S.G. VAN DER WAL

COMPUTER PRESTATIE ANALYSE

BASISRELATIES



ACADEMIC SERVICE

COMPUTER PRESTATIE ANALYSE

S.G. VAN DER WAL

COMPUTER PRESTATIE ANALYSE

BASISRELATIES


ACADEMIC SERVICE

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Wal, S.G. van der

Computerprestatieanalyse : basisrelaties / S.G. van der Wal. -
Schoonhoven: Academic Service
Met lit. opg.
ISBN 90-6233-250-1
SISO 520.5 SVS 8.12.3 UDC 681.3 NUGI 852
Trefw.: informatica.

10 9 8 7 6 5 4 3 2 1

Uitgegeven door: Academic Service
Postbus 81
2870 AB Schoonhoven

Zetwerk: het boek is door de auteur gezet in Troff en camera-ready aangeleverd. De tekst is gezet uit de Times Roman en belicht bij het Centrum voor Wiskunde en Informatica, Amsterdam.
Omslag illustratie: Anna Vollema; uitvoering Prographics, Bussum.
Druk: Krips Repro Meppel.
Bindwerk: Meeuwis, Amsterdam.

ISBN 90 6233 250 1
NUGI 852

Inhoud

Woord vooraf

Dit boek is gegroeid uit de cursus 'Computer-prestatieanalyse', die deel uitmaakt van het curriculum van de HIO-ENSCHEDE, de opleiding HOGER INFORMATICA ONDERWIJS aan de Hogeschool Enschede.

De kern van het boek wordt gevormd door de hoofdstukken 4 tot en met 10 en 12 tot en met 16. Deze komen in de eerste twee modulen van de cursus aan de orde. In de derde module worden methoden om separabele queuing-netwerken door te rekenen behandeld.

Het is een groot genoegen mijn oud-collega JOS VAN DER MEER dank te zeggen voor zijn bijdrage aan de voorbeelden en opgaven.

Graag wil ik RONNY AKKERSDIJK, HENK VAN HET BOLSCHER, A.J. VAN VEEN, JAAP VERMEULEN, ANNA VOLLEMA, FRANS VOOGD en JOS VAN VROONHOVEN bedanken voor hun hulp. FRANK KUIPER en DAAN OTTEN waren steun en toeverlaat op het CWI.

De informatie die in de loop van de jaren naar voren kwam in de bijeenkomsten van de werkgroep 'Computer Prestatie Evaluatie' van het ASI/NGI is dankbaar verwerkt.

Het boek is door de auteur gemaakt en gezet met TROFF, het tekstverwerkingssysteem onder UNIX uit de tweede generatie. Het Centrum voor Wiskunde en Informatica (CWI) in Amsterdam verleende hiervoor gastvrijheid.

Enschede/Bedum, Siebren van der Wal

1

Inleiding

1.1. HISTORISCH OVERZICHT

Lang, heel lang, was de computer door mythen omhuld, als een tovertuig van een hoge statuswaarde. Wie er geregeld mee omging, sprak daarover in een versluierend jargon.

Het management van een bedrijf dat een computer bezat, verwachtte veel —zo niet alles— van het vreemde instrument, maar het huiverde daaruit consequenties te trekken. Het gaf geen schok als verwachtingen niet uitkwamen en voorstellen voor wijziging van de kostbare apparatuur werden zonder aarzelen gefiatteerd.

De laatste jaren is het aanzien snel veranderd. Computers zijn alledaags geworden en hebben hun glans verloren, ze worden aangeschaft en vervangen zodra dat voor het bedrijf aantoonbaar nut oplevert.

Tijdens de economische regressie werden managers gedwongen om bij elke mooie nieuwe computer na te gaan, welke voordelen de aanschaf er van bood. Zo hebben ze ontdekt dat het om een gewoon elektronisch instrument gaat. Bij zo'n instrument hoort, net als bij elke andere produktiemachine, een kosten-baten analyse en een periodieke rapportering over de kwaliteit van de verwerking. Iedere voorgestelde uitbreiding zal daarom vergezeld moeten gaan van een uitspraak over de produktiviteitsverbetering. Ook bij computers wordt tegenwoordig verwacht dat er voortdurend wordt geanalyseerd hoe de huidige machine presteert en op welk toekomstig presteren mag worden gerekend.

De beheerders van computers, de systeemontwerpers en -analisten blijken wel eens niet goed voorbereid om zulke vragen te beantwoorden. Ook zij hebben soms lang met een vast idee geleefd: ze wisten dat een computer intern ingewikkeld funktioneert en meenden dat deze zich daarom onttrekt aan elke poging het gedrag er van nauwkeurig te voorspellen. Als een configuratie volloopt zou er eigenlijk maar één manier zijn om de problemen de baas te worden: schaf een snellere computer

aan; een snellere processor levert een overeenkomstig hogere produktie omdat de fabrikant de rest van de configuratie passend meelevert.

Toch moet ook deze onvoorspelbaarheid en onberekenbaarheid een mythe zijn. In andere vakgebieden funktioneren zaken ook heel complex als er in detail wordt gekeken. Maar dan zijn stellig algemene globale uitspraken mogelijk. In de natuurkunde bijvoorbeeld hoeft niet de precieze loop van de moleculen bekend te zijn om te kunnen voorspellen hoe de druk verandert als een gas uitzet.

Ieder onderzoek dat zich bezig houdt met het presteren, met de 'performance', van een systeem wordt in het algemeen prestatieonderzoek of performance-analyse genoemd; performance en prestatie zijn synoniem. Bij systemen als computerconfiguraties of computernetwerken wordt gesproken van 'computer performance analysis' of computer-prestatieanalyse†. Dit bestudeert het funktioneren van computerconfiguraties en -netwerken en is gericht op het vergelijken van de verwerking in de huidige situatie met die onder gewijzigde omstandigheden na eventuele wijzigingen.

Computer-prestatieanalyse is steeds duidelijker nodig. Gelukkig ontwikkelt deze nog jonge informatica-discipline zich erg voorspoedig. Aanvankelijk werd het vakgebied beoefend door twee heel verschillende groeperingen: enerzijds computerbeheerders, vol vertrouwen in oppervlakkige intuïtieve argumenten — anderzijds doorgewinterde mathematici, vastbesloten het gedrag van een computersysteem in formules te vangen. Later zijn practici en theoretici meer naar elkaar toegegroeid: intuïtieve argumenten kregen een duidelijke harde achtergrond en mathematische argumenten werden rechtstreeks in verband gebracht met het waargenomen gedrag van computersystemen.

Dit gebeurde vooral nadat rond 1976 was vastgelegd onder welke mathematische voorwaarden de kansen op de toestanden van een systeem door de 'produktvorm' van BASKETT et al. worden aangegeven. De toenadering werd versneld door de geruchtmakende aanpak van J.P. BUZEN en P.J. DENNING, die door hen 'operational analysis' is gedoopt. In deze operationele analyse wordt geprobeerd een rechtstreekse koppeling te bewaren tussen mathematische en meetbare grootheden. De blijvende winst van deze methodiek zal liggen in de zorg voor het zichtbaar blijven van de verbinding tussen de mathematisch/stochastische veronderstellingen en de aan een systeem meetbare grootheden.

In het begin van de tachtiger jaren kwam de kunst van het modelleren tot grote bloei door het ontwikkelen van doorzichtige algoritmen om de 'produktvorm'-oplossing van separabele queuing-netwerken door te rekenen. Vooral het MVA-algoritme van M. REISER blijkt een bereik te hebben, dat zich met passende benaderingen uitstrekt tot ver voorbij de oorspronkelijke context.

De nu bereikte resultaten aan inzichten en technieken zijn indrukwekkend. DENNING, die ook op andere disciplines uit de informatica (computer science) baanbrekend werk heeft verricht, noemde eens —in zijn funktie van voorzitter van de

† Ook Engels gespeld als Computer Prestatie Analyse.

ACM— computer performance 'experimental computer science at its best'. Toch zijn nog lang niet voor alle problemen analyses beschikbaar en de onstuimige groei en de snelle veranderingen in de computerwereld roepen steeds nieuwe vragen op.

1.2. INLEIDING

Met deze 'computer performance' gaan we ons bezig houden. Hopelijk wordt duidelijk dat de mythe van de onvoorspelbaarheid het licht van de rede niet verdraagt: ook hier kunnen met simpele hulpmiddelen zinvolle voorspellingen worden gedaan. De belangstelling gaat uit naar eenvoudige algemene redeneringen, niet naar ingewikkelde specifieke verhandelingen. Algemene inzichten zullen immers hun waarde behouden, hoe de computers en de computerwereld ook mogen veranderen; ze maken het zowel beheerders als systeemontwerpers mogelijk snel verwachtingen aan te geven.

De mathematische hulpmiddelen die gebruikt worden zullen nergens geavanceerd zijn, ze gaan niet uit boven eenvoudige analyse en algebra. Wel zullen statistische begrippen en zienswijzen van belang zijn. Op een computersysteem wordt steeds een wisselend beroep gedaan; bij de uitvoering van taken treden telkens andere volgorden van bewerkingen op. Alle aan een computersysteem te meten grootheden worden dan ook gezien als stochasten, met een kansverdeling en een verwachtingswaarde. Kennis van statistische analyse is nodig om iets zinnigs te kunnen gaan zeggen wanneer er iets gemeten is.

Meestal zullen onze voorspellingen slaan op de verwachtingswaarde (het 'echte' gemiddelde) en dat blijkt een behoorlijk 'robuuste' grootheid te zijn, die weinig verandert bij een nadere precisering van de situatie. Heel veel belangrijke globale redeneringen zeggen alleen iets over verwachtingswaarden. In de praktijk is vaak ook informatie over de spreiding rond het gemiddelde nodig om de kwaliteit van de service aan te geven. Voorspellingen daarover zijn meestal enkele klassen moeilijker en dikwijls zelfs niet te geven. Wel behandelen we gemiddelden voor deelgroepen uit de hele verdeling.

Als kader voor de aanpak wordt zowel de 'stochastische' als de 'operationele' weg benut. We geven de operationele aanpak een eigen interpretatie, waarbij het verband met de stochastische zienswijze duidelijk is. De charme van het operationele is dat de essentiële samenhang helder wordt weergegeven. Ernaast staat steeds de 'klassieke' stochastische benadering volgens bijvoorbeeld de theorie van Markovketens en Markovprocessen. De nadruk, die in het begin ligt bij de operationele aanpak, wordt allengs verschoven naar de stochastische zienswijze, omdat die toch een rijkere theorie oplevert.

Om het praktisch belang van het betoog te onderstrepen worden altijd in het echt opererende systemen en onderdelen daarvan als voorbeelden gebruikt. Er wordt steeds —om zo te zeggen— een model voor zo'n systeem geanalyseerd. Zo'n model zal in eerste instantie eenvoudig zijn, maar kan vaak zonder aan de essenties afbreuk te doen worden uitgebreid. In een model zijn alleen de voor de vraagstelling relevante specifikaties ingebracht.

Simulatie wordt veel gebruikt om de eigenschappen van specifieke systemen na te bootsen. Omdat we uit zijn op algemene inzichten komt de simulatietechniek hier niet aan de orde, ze ligt buiten het bestek.

1.3. TOEPASSINGSGEBIEDEN VAN COMPUTER-PERFORMANCE

Een inefficiënt computersysteem is altijd min of meer een ramp. De toepassingsgebieden van computer-performance liggen dan ook bij:

- het tunen van computers.

Het operatingsysteem bevat vrijwel altijd een aantal parameters die door de beheerder worden ingesteld. Een computer kan dan ook goed of slecht zijn afgesteld (getuned). Het is als bij het uitlijnen van een zeilboot: een precieze berekening volgens de wetten van de aerodynamica is vrijwel nooit nodig, maar wel kennis van de basisprincipes van dat vak. Of als bij het bouwen van een kathedraal: de wetten van de statica moeten in acht genomen worden.

- het aangeven van de in de toekomst benodigde systeemcapaciteit (capaciteitsplanning).
- het aangeven van de optimale grootte van een configuratie (sizing).

De installatie zal de werklast met minimale totale kosten moeten verwerken. De aanschaf-, onderhouds- en afschrijvingskosten moeten in rekening worden gebracht naast andere kostenfaktoren, zoals de wachttijden van gebruikers.

- het eerlijk verdelen van de computerkosten over de gebruikers.

Op vrijwel ieder systeem wordt 'accounting' bijgehouden; dit zijn gegevens zoals CPU-tijden en diskaccessen van gedraaide jobs. Op basis van deze accountinggegevens moeten aan de klanten de kosten van hun jobs in rekening worden gebracht. Meestal moet deze toerekening zo 'eerlijk' mogelijk geschieden. Klanten zitten elkaar in de weg bij het verdelen van de schaarse hulpmiddelen. In welke mate beïnvloeden bepaalde klanten de prestaties van het systeem?

- het bewaken van de kwaliteit van de service.

Gebruikers wordt een bepaalde service contractueel gegarandeerd. Waardoor wordt zo'n norm niet gehaald?

- het opstellen van kwaliteitsnormen voor computers.

Een computer wordt gezien als een machine die bepaalde taken verricht. Die taken kan hij goed of slecht verrichten. Ook bij de zeer ingewikkelde Hi-Fi apparatuur zijn DIN-normen voor kwaliteit opgesteld!

- het ontwerpen en begrijpen van operatingsystemen.

De beschrijving van een operatingsysteem bestaat meestal uit source code voorzien van enig commentaar. Verder zijn in een begeleidend rapport de basisideeën aangegeven. Er blijkt bij bestudering steeds dat de ontwerpers eenvoudige elementaire redeneringen als leidraad hebben gebruikt. Aan elk operatingsysteem ligt een filosofie ten grondslag over de gewenste onderlinge beïnvloeding van processen die om hulp en steun van de verwerkingseenheden dingen. Om ongewenste situaties te voorkomen worden door het operatingsysteem maatregelen genomen. Ook om de volgende, niet direkt voor de hand liggende, redenen is een prestatie-analyse van belang:

1) Als een operatingsysteem wordt bestudeerd vanuit zo'n gezichtspunt is men er op gespitst uit te vinden welke volgorden van gebeurtenissen vaak en welke weinig voorkomen, welke relevant zijn voor de verwerking en welke niet. Men ontdekt dan heel wat weinig gebruikte uitgangen en akties voor 'in geval van nood', die worden uitgevoerd in de tijd dat er eigenlijk geen werk is of als er te veel werk is. Als resultaat blijft een kern over van normale verwerking, die relatief gemakkelijk te beschrijven, te doorzien en te hanteren valt.

2) De eenvoudige redeneringen die aan het operatingsysteem ten grondslag liggen zijn vaak gebaseerd op 'intelligente intuïties' van de ontwerper (bijvoorbeeld geef bij de CPU altijd I/O-intensieve programma's voorrang). Het is mogelijk deze intuïties te toetsen en hun beperkingen te doorzien door ze voorspellingen te laten doen in situaties die helemaal doorgerekend kunnen worden. We zullen heel wat situaties vercijferen en steeds kijken wat 'intuïtie' zou zeggen. De voorzorgen van een ontwerper kunnen zo geëvalueerd worden.

Verdere toepassingsgebieden zijn natuurlijk:

- het ontwikkelen van nieuwe computers, computerconfiguraties, caches, netwerken en protocollen van netwerken.
- enzovoort.

Van belang is steeds dat intuïties kunnen worden getoetst. Na berekeningen zijn de relatieve grootten van effekten bekend en is duidelijk wat hoofdeffekten en wat bijzaken ('hogere orde effekten') zijn. Heel veel intuïtie is wel in staat om het bestaan van een effekt te verklaren, maar geeft volstrekt niet de relatieve grootte van het effekt aan —hoe helder en duidelijk de intuïtie ook is verwoord. Door het opbouwen van deze 'reken'-ervaring ontstaan meer doordachte 'intuïties'.

Soms wordt gedacht dat het doel van computer-prestatieanalyse zou zijn een bepaald operatingsysteem of een heel bepaalde configuratie in alle detail door te rekenen. De hoeveelheid informatie die nodig is om zo gedetailleerd voorspellingen te doen is echter frustrerend groot en de opbrengst in kwaliteit van de voorspelling is vaak niet lonend.

Tegenwoordig zijn er goede programmapakketten in de handel, waarmee zo'n taak aanzienlijk vereenvoudigd kan worden; ja, eigenlijk voor het eerst economisch mogelijk wordt. Ook dan zal het nut van de uitkomsten in de eerste plaats bestaan uit het scherpen van het inzicht in het funktioneren en presteren van het systeem.

1.4. GESCHRIFT

Het is in dit boek vooral de bedoeling om redeneringen te onderbouwen, die vanuit de dagelijkse ervaring als 'gezond verstand' naar voren worden gebracht bij het waarderen van wat een computer presteert. Het zal gaan om de *basisbetrekkingen* tussen grootheden, die de performance beknopt beschrijven.

We willen laten zien dat het verkeerd is te denken dat de veronderstellingen en redeneringen, die gebruikt worden bij het vinden van 'mathematische formules' voor prestatiegrootheden, anders zouden zijn dan de veronderstellingen en redeneringen die nodig zijn om de alledaagse computerprestaties te bespreken. Er is geen gradueel verschil en geen essentieel verschil! Daarom zullen we nooit (of vrijwel nooit) formules zonder afleiding presenteren. Zonder inzicht in de achtergronden —en dat is in de afleiding en de bijbehorende veronderstellingen— is het inbrengen van formules als het aanbrengen van patches in computercode: gezegend zij de greep. De mathematische stappen die dwingend volgen uit de gemaakte veronderstellingen zullen we niet steeds in detail weergeven, daar ligt het zwaartepunt niet. Maar er ontstaat pas een eenheid tussen ervaring, inzicht en beschrijving als er een gemeenschappelijke basis is.

Alle informatie die tot de bagage van gevestigde disciplines behoort zullen we aan deze vakgebieden ontlenen en niet opnieuw ontwikkelen. We verwijzen algemeen naar handboeken op het gebied van computer-architectuur, netwerken, operatingsystemen, statistiek, wiskunde, enzovoort.

In dit boek gaan we een aantal basisrelaties van alle kanten bekijken. Deze relaties hebben een universeel karakter, we zullen zien dat ze op allerlei plaatsen onder heel verschillende vormen opduiken.

Het zijn de relatie van Little (hoofdstuk 4) en de responsetijdrelatie (hoofdstuk 5), het verband tussen de kans dat een toestand optreedt en de kans dat de toestand wordt aangetroffen (hoofdstuk 7), de beschrijving van toestandsovergangen zonder op de invloed van een lang verleden te letten: de Markovketens (hoofdstuk 9), en de schat aan informatie die beschikbaar is als gebeurtenissen lukraak plaats vinden: het Poissonproces (hoofdstuk 12).

Dit levert een verzameling inzichten en gereedschappen op, die èn bij alledaagse filosofieën nodig zijn èn in de wachttijdentheorie bij computermodellen. In de

laatste hoofdstukken gebruiken we deze basisrelaties om de gemiddelde responsetijd in de bekendste klassieke wachttijdensystemen te berekenen.

1.5. WOORDKEUS

In plaats van het in de statistische literatuur ingeburgerde 'random' gebruiken we consequent het beeldende *lukraak*. Theorie en praktijk van de computerprestatieanalyse zijn vol met gevolgen van dat er iets lukraak of 'bijna lukraak' gebeurt; we gaan leren wat de kracht is van het toverwoord 'lukraak'. Soms lijkt het alsof lukraak gradaties zou kennen, zo spreken we van 'zuiver lukraak'.

Het Engelse begrip rate voor onder andere 'aantal per tijdseenheid' zullen we vaak vertalen met *'snelheid'*; als er drie signalen per tijdseenheid binnenkomen zeggen we dat de snelheid waarmee de signalen binnenkomen drie per tijdseenheid is. Op andere plaatsen vertalen we het begrip 'rate' met stroom of frequentie van optreden.

De tijdsduur tussen het begin en het eind van een toestand noemen we de *duur* van de toestand. We zullen dus bijvoorbeeld spreken over de verdeling van de duur van een toestand. De tijdsduur tussen het moment waarop er iets gebeurt en het eerstvolgende moment waarop iets soortgelijks gebeurt noemen we een *tussenpoos*. We willen zoveel mogelijk vermijden dat er verwarring ontstaat door de dubbele betekenis van tijd: als tijdstip en als tijdsduur.

Dat is ook de reden dat we, als we wat formeler worden, bij voorkeur de uitgang -duur in plaats van -tijd gebruiken voor de tijd dat iets duurt. De (stochast) wachtduur geeft aan hoelang de wachttijd is. Maar we zijn niet zo consequent dat we over responseduur in plaats van responsetijd spreken.

In ons begrip *werk*, en daarmee in *werklast*, zit alleen het aspekt 'wat' en niet het aspekt 'hoe vaak per tijdseenheid'. Dit laatste vangen we onder 'drukte', dus als het drukker wordt verandert de samenstelling van de werklast (in eerste instantie) niet. Natuurlijk neemt bij toenemende drukte de last, die de machine wordt opgelegd, wel toe.

Een *server* is een 'bediende' die opdrachten van klanten uitvoert. Er is een zee van aanduidingen —load, service demand, service time, enzovoort— voor de tijd waarin een server een klant helpt. We zullen in het algemeen spreken van *verwerkingsduur* en als we ons specialiseren tot de wachttijdentheorie† (met een server die steeds even snel werkt) spreken we van *serviceduur*. De bienduren zijn de tussenpozen waarmee klanten die afgewerkt zijn, de server verlaten. Een server die geen werk heeft is werkeloos of *'idle'*. De *bezettingsgraad* van de server geeft aan welk deel van de tijd deze bezig is. De afwikkeling van het werk van klanten door een server kan op vele manieren gebeuren; de bekendste regelingen staan in §14.10.

---

† Queuing theory wordt vertaald met wachttijdentheorie.

Een *computerconfiguratie* of kortweg configuratie is opgebouwd uit processoren (CPU's) en randapparaten; configuratie is een algemeen begrip, het kan een mainframe zijn of een mini, maar ook een gedistribueerd lokaal netwerk met werkstations. Een verwerkingseenheid zoals die zich in de systeemsoftware manifesteert, noemen we een *device*, dit kan een 'aangekleed' randapparaat zijn, of een CPU. Als randapparaten zullen we meestal alleen de disks onderscheiden, wanneer we meer verschil willen aangeven voeren we daarnaast een 'drum' op; drum is dus de tegenhanger van disk. Zo'n 'drum' zal soms een diskette zijn of een solid state device, of zelfs een remote server.

Een deel van de gebruikers benut de gemeenschappelijke middelen door afwisselend te vragen om en te reageren op de verwerking van een opdracht. Omdat terminal-gebruik het prototype van deze interaktie is, duiden we de betrokken gebruikers aan als *terminalisten*.

Verder is UNIX een handelsmerk† van BELL LABS.

1.6. NOTATIE

We drukken de tijd uit in tijdseenheden (t.e.). *Operationele schatters* geven we aan met een aanhalingsteken. Verwachtingswaarde en variantie worden aangegeven met E en VAR, variantie is een maat voor de *spreiding*. Pr staat voor 'kans op' (probability).

Een bepaalde toestand, zeg toestand A, wordt ook als toestand$_A$ genoteerd.

Opgaven en formules zijn per hoofdstuk genummerd. Naar de formule met nummer (1.1) wordt verwezen als (1.1), naar de opgave met nummer (1.1) als opgave 1.1. In de titel van een paragraaf of opgave kan gerefereerd worden aan andere paragrafen, de verwijzing naar §1.1.1 wordt dan geschreven als 1.1.1. De voorbeelden en opgaven zijn in het register opgenomen.

De intermezzo's geven verdieping en verbreding, maar bevatten niet de hoofdlijnen van de stof. Antwoorden van opgaven zijn dikwijls tussen haakjes toegevoegd.

---

† Unix is a Trademark of AT&T Bell Laboratories.

2

Achtergronden

2.1. INLEIDING

Bij het beheer van een computerconfiguratie wordt de prestatie-analyse bijna dagelijks toegepast. Als schets van een van de achtergronden van het vakgebied gaan we in op de rol, die het op deze specifieke plaats speelt.

2.2. PERFORMANCE VAN EEN CONFIGURATIE

Het moderne beheer van een computerconfiguratie is niet denkbaar zonder goede faciliteiten om vast te houden wat er met de machine gebeurt. Op zulke gegevens zal elke performance-analyse stoelen.

De schat aan informatie die in principe kan worden verzameld, moet worden gereduceerd tot hanteerbare overzichten. Er zal steeds veel datareduktie worden toegepast, het vak statistiek levert daarvoor hulpmiddelen. De prestatiegegevens moeten voor een deel on-line beschikbaar zijn omdat ze een dynamisch karakter hebben, zoals waarden voor de huidige responsetijd of de momentane bezettingsgraad van kanalen. De statistisch bewerkte informatie zal in een database worden opgeslagen, die interaktief kan worden geraadpleegd.

Een frustrerende ervaring bij deze vastleggende taak is dat het vaak lijkt of zelfs de op het oog meest simpele zaken niet strikt bepaald kunnen worden. Dit geldt al voor begrippen als werklast en snelheid. In het begrip werklast wordt vagelijk aangegeven wat er op de machine zoal verwerkt wordt, en hoeveel er verwerkt wordt. De snelheid van de configuratie bepaalt hoelang de machine over die verwerking doet. Over werklast en snelheid zullen we veel moeten zeggen, het onderlinge verband is niet triviaal.

Een van de grote voordelen van het ontwikkelen van modellen is trouwens dat veel duidelijker wordt welke informatie vastgehouden moet worden. We gaan direkt maar de eerste confrontatie met werklast en snelheid aan.

2.2.1. snelheid en mips

Op de vraag hoe snel de machine is, geeft de fabrikant een antwoord uitgedrukt in mips, het getal geeft aan hoeveel miljoen instrukties er per sekonde uitgevoerd worden als de CPU aktief is (*mips* = million instructions per second). De waarden variëren van 0.5 mips voor mini's tot honderd(en) mips voor de echte grote mainframes. Deze mips-snelheid zegt dus alleen iets over de CPU, niets over de snelheid van de randapparaten of van de configuratie als geheel.

Welke 5 millioen instrukties worden er per sekonde uitgevoerd bij een snelheid van 5 mips? De opgegeven mipsrate veronderstelt een bepaalde verdeling van het aanbod aan instrukties over de mogelijkheden: een mix met relatief zoveel Moves, relatief zoveel Register-Register instrukties, relatief zoveel 3-adres instrukties, enzovoort. De gebruikte mix is door de fabrikant samengesteld uit metingen bij zijn klanten, of ontleend aan een standaardmix, die bijvoorbeeld ooit ontwikkeld is ten behoeve van onderzoek naar de kwaliteit van software voor wetenschappelijk rekenwerk. Zo'n mix zal hoogstwaarschijnlijk niet overeenkomen met de verdeling van de instrukties in de huidige door de machine verwerkte werklast. Relevant kan hij alleen zijn als de samenstelling van de te draaien werklast een grote overlap heeft met wat de fabrikant als gemiddeld in zijn klantenkring ervaart. De opgegeven mipsrates van machines van verschillende fabrikanten of —zelfs bij dezelfde fabrikant— van verschillend type processor zullen dan ook meestal niet zonder meer te vergelijken zijn. Voor een goede interpretatie van de mipsrate is eigenlijk altijd kennis van de eigen werklast nodig. Bekend is het geval waarin het gebruik van een optimaliserende COBOL compiler ineens geen zin meer bleek te hebben: op de na een 'upgrade' aangepaste snellere (hogere mipsrate) processor was de met deze compiler verkregen code langzamer dan de code gegenereerd door de standaard COBOL compiler. Een optimaliserende compiler probeert bij alternatieven voor de te genereren code altijd op basis van de verwerkingstijden van instrukties de qua executietijd kortste sequentie van instrukties te leveren. Bij het upgraden van de CPU worden natuurlijk niet alle instrukties in dezelfde mate versneld, hierdoor kunnen sequenties die eerst tijdwinst betekenen na de upgrade alle aantrekkelijkheid verloren hebben. Dat was hier het geval. Vergroting van de mipsrate met een bepaalde factor wil niet zeggen dat alle instrukties zoveel sneller zijn geworden!

Het begrip mipsrate voorveronderstelt dat een bepaalde instruktie een zekere gemiddelde relatieve verwerkingstijd heeft. Dat zal echter op veel computers thans niet meer het geval zijn. Voor de moderne CPU's (ALU's) met een sterke interne parallelliteit op het niveau van de microcode (pipelined) zal de verwerking van een bepaalde instruktie afhangen van de toevallig voorgaande en volgende instruktie. Immers delen van de huidige instruktie zullen zoveel mogelijk parallel in de tijd met delen van de voorgaande en volgende instrukties worden afgehandeld. Daardoor ontstaat een grote correlatie tussen opeenvolgende verwerkingsduren. Ook het gebruik van een instruktiecache tussen processor en geheugen verstoort volledig het gebruikelijke beeld: voor instrukties die 'toevallig' in de cache zitten wordt de executieduur sterk gereduceerd.

Het wordt dus steeds minder zinvol om over de relatieve snelheid van de ene ten opzichte van de andere instruktie te spreken en daarmee wordt het begrip mipsrate nog minder grijpbaar. De mipsrate geeft —op zijn best, en dat is zijn kracht— een relatieve snelheid van de processor aan. Zelfs op dit laagste niveau van vergelijking —per instruktie— blijkt het niet mogelijk *hoe snel wordt iets verwerkt* los te koppelen van *wat wordt verwerkt*, en zijn snelheid en werklast afhankelijk!

2.2.2. *snelheid taalgericht*

Door de redakties van computermagazines worden vaak snelheidsvergelijkingen gepubliceerd. Er worden daarbij een aantal programma's in een bepaalde taal gedraaid op alle computers met een compiler voor die taal. De draaitijden geven dan de relatieve snelheden.

Bij zo'n 'vergelijking op het hoogste niveau' worden alle verschillen op alle niveaus meegenomen. De verschillen tussen compiler, operatingsysteem, implementatie van het stackmechanisme, van I/O, enzovoort maken uiteindelijk het verschil in draaitijd uit voor deze heel specifieke applicatie. Door alleen de draaitijden te vermelden worden de gewichten van deze verschillen helemaal niet gedemonstreerd. Voor iemand die op een van deze 'gemeten' machines gaat werken, blijft het de vraag wat de diverse verschilpunten betekenen voor de verwerkingssnelheid van zijn werklast.

2.2.3. *werklast en benchmark*

Met het begrip werklast wordt aangegeven welke programma's er verwerkt worden, en hoe vaak die programma's afgewerkt worden. Maar hoeveel verschil is er niet in de programma's die door een configuratie behandeld worden!

Natuurlijk probeert men te komen tot een indeling naar 'soort' werk. Het gaat er om het meest voorkomende werk te karakteriseren en slechts dat op te voeren. Statistische technieken als cluster-analyse helpen hierbij, maar veel inspanning is nodig om verder te komen dan een constatering als: gemiddeld zoveel COBOL compilaties, zoveel PASCAL compilaties, zoveel executeerbare programma's en zoveel edit-werk.

Om de snelheid, waarmee de werklast verwerkt wordt, te meten wordt vaak een 'representatieve' verzameling programma's —een *benchmark*— samengesteld uit de programma's die veel worden gedraaid. De verwerkingssnelheid wordt geschat met de inverse van de tijdsduur, waarin deze benchmark in zijn geheel verwerkt wordt. Die tijd kan simpel bepaald worden door de programma's uit de benchmark tijdens hun verwerking te timen. Maar zullen die programma's parallel gedraaid worden, door ze bijvoorbeeld tegelijk te laten starten, of komen ze sequentieel aan bod? Bij het eerste worden ze onder multiprogrammering verwerkt, en dat is realistisch, maar de multiprogrammeringsgraad zal dalen naarmate er meer gereed zijn. En wordt de benchmark verwerkt op een representatief moment waarop ook de normale gebruikers aktief zijn, of op een moment dat er verder niets loopt?

Het criterium om hiertussen te beslissen behoort natuurlijk te zijn dat de situatie met een draaiende benchmark zo volledig mogelijk overeenstemt met de situatie

waarin de configuratie gemiddeld verkeert als de echte werklast draait. De overeenstemming zal er niet alleen moeten zijn voor de bezetting van de CPU('s), maar ook voor de bezetting van disks, netwerkdrivers, kanalen en andere randapparaten. En ook voor de interface met terminals!

Een interaktieve inbreng kan gesimuleerd worden door personen opdrachten uit een voorgeschreven script te laten inbrengen, maar dat is duur. Het is ook mogelijk de simulatie te laten verzorgen door extra systeemprogrammatuur; de benodigde systeemroutines worden vaak door de leverancier bijgeleverd. Er worden dan processen gegenereerd, die zich als terminalisten gedragen. Deze software-simulatie betekent echter een verandering van de systeemsoftware ten opzichte van de normale situatie —en dus een verstoring van het te meten systeem door het meetinstrument. En zoiets behoort bij elke meting zoveel mogelijk te worden vermeden. De verstoring is er niet als op de terminalpoorten microcomputers worden aangesloten, waarop programmatuur loopt die opdrachten genereert. Deze 'externe' simulatie is alleen praktisch uitvoerbaar als goedkope portable micro's worden genomen, die bijvoorbeeld via een apart lokaal net geladen en uitgelezen worden.

Het ideaal dat het draaien van de benchmark de normale aktiviteit op het systeem keurig weerspiegelt, is moeilijk te benaderen met een benchmark opgebouwd uit geselekteerde 'veel gedraaide' programmatuur. Het is vooral lastig te garanderen dat de resultaten representatief zijn. Eigenlijk kan dit alleen maar lukken als in de programmatuur, waaruit de benchmark is opgebouwd, instelbare parameters zijn opgenomen. Die moeten dan proefondervindelijk op zulke waarden worden getuned, dat de overeenstemming zo goed is als maar haalbaar. Zulke parameters zijn bijvoorbeeld het relatieve optreden van stukken programmatuur die meer CPU-tijd dan I/O-tijd vergen —die *CPU-bound* zijn— en stukken waarvoor het omgekeerde geldt, die *I/O-bound* zijn. Een benchmark samengesteld uit kleine stukken programmatuur met elk een eigen karakteristieke belasting van een resource, waarbij de samenstelling via parameters beheerst kan worden, heet een *synthetische benchmark*. Zo'n benchmark kan een krachtig hulpmiddel zijn.

2.3. MONITORS

2.3.1. hardware-monitors

Het verzamelen van gegevens over het funktioneren van de machine heet in het vakjargon van de systeembeheerder *monitoren*, het is hetzelfde als meten. Voor dit monitoren zijn twee wezenlijk verschillende hulpmiddelen beschikbaar: hardware-monitors en software-monitors.

Een hardware-monitor tapt electrische signalen af en registreert deze. Op de interessante data- en adreslijnen van het te meten object worden met probes (klemmen) aansluitingen aangebracht, de door sensoren opgevangen signalen worden via kabels naar een minicomputer geleid en daar verder verwerkt. Deze minicomputer is meestal voorzien van extra hardware als tellers, poorten, clock, enzovoort. Het

grote voordeel van de hardware-monitor is dat hij zelf geen belasting voor het systeem betekent. Dit meetinstrument beïnvloedt het te meten instrument niet, omdat het zelfstandig werkt op eigen kracht. Het grote nadeel is dat altijd anonieme informatie wordt verkregen: er wordt een signaal geconstateerd, maar wie of wat dit signaal verwekte blijft onbekend.

In principe kan met een probe 'op de processor' worden nagegaan wanneer de processor switcht, dus van het ene proces op het andere overstapt. Maar er kan niet worden bepaald of het volgende proces in executie wordt genomen omdat het vorige een I/O-opdracht genereerde, of dat de time slice vol was of dat er een page fault optrad. Hardware-monitors worden voornamelijk toegepast op plaatsen waar anonieme informatie zonder meer relevant is, bijvoorbeeld bij het meten van de verdeling van accessen over de kanalen van een groter randapparaat, het verkeer over een kabel in een netwerk, de hitrate van een cache of de responsetijd aan een terminal.

Voordat de hardware-monitor kan worden ingezet moet eerst via de hardware-schema's van het te meten apparaat worden uitgevonden langs welke uitgangen de interessante signalen naar buiten komen, bijvoorbeeld op welke pennen van de backplane van de computer. Op die plaatsen moet dan een probe bevestigd worden. Door het oprukken van de chips wordt het steeds lastiger aansluitpunten te vinden. Chips worden geïntegreerd tot 'multiple chip carriers'; alle interessante data- en adreslijnen blijven, om de verbindingen kort en dus snel te houden, binnen deze carriers en worden onbereikbaar. Het aansluiten van de probes betekent eigenlijk steeds een grote potentiële bron van storingen, vakkundige hulp —ook van de fabrikant— is geboden.

De laatste jaren zijn hardware-monitors voor het globale systeembeheer aan het uitsterven, de oorzaken van hun verdwijnen zijn waarschijnlijk: ze zijn prijzig, aan het gebruik kleven grote praktische bezwaren, ze leveren vaak niet de adequate informatie en op ieder systeem is tegenwoordig wel een redelijke software-monitor beschikbaar.

2.3.2. software-monitors

Een software-monitor maakt gebruik van 'tellers', dat zijn records met informatie, die bijgewerkt worden als een stuk code doorlopen wordt (de interceptietechniek), of die periodiek worden bijgesteld, bijvoorbeeld bij binnenkomen van een clockinterrupt (de bemonsteringstechniek, §7.6). Zo'n monitor is een gestructureerde verzameling van routines, die zorgen voor dit bijwerken. Vaak wordt daarbij gebruik gemaakt van systemcalls, dat zijn aanroepen van funkties uit het operatingsysteem. Een software-monitor wordt meestal door de leverancier van het operatingsysteem meegeleverd.

Het nadeel van een dergelijke monitor is dat de normale gang van zaken wordt beïnvloed, omdat er voor het meten gebruik moet worden gemaakt van de beschikbare middelen van het te meten systeem. De verstoring kan gelukkig meestal tot zo'n 5% beperkt blijven. De monitor verzorgt dan strikt alleen het eigenlijke verzamelen, alle verwerking vindt elders achteraf plaats. Heel vaak loopt de monitor

standaard mee ten behoeve van het systeembeheer, dan vervalt het bezwaar dus min of meer.

Een eenvoudige maar erg efficiënte monitor kan worden verkregen door gegevens die de processor toch al verzamelt, periodiek uit te lezen; meestal wordt er door de processor een historie bijgehouden om in geval van nood recente informatie te hebben over de stand van de uitgevoerde instrukties. Zo heeft bijvoorbeeld PAANS metingen gedaan met een monitor, die de 'in core Trace Table' uit het operatingsysteem MVS periodiek uitleest. De informatie uit deze tabel wordt opgebouwd door de microcode uit de CPU.

De informatie die door een software-monitor wordt bijgehouden, kan gericht zijn op het resourcegebruik door de gebruikers. Zo kan van elke terminalist worden geteld welke commando's hij startte, welke files hij gebruikte, wanneer hij probeerde beschermde informatie te lezen, hoe vaak hij welke disks benaderde, enzovoort. Zulke gegevens zijn van belang als men gebruikers wil belasten naar het specifieke beslag dat ze op de machine leggen, op grond van dit type informatie kan immers een gespecificeerde rekening worden samengesteld. Dit zijn de *accounting* gegevens (of gegevens voor de doorberekening), algemeen gezien zou men daaronder alle informatie kunnen verstaan, die aan gebruikers gekoppeld is.

De software-monitor kan ook algemene 'overall' gegevens verzamelen over het resourcegebruik door de lopende processen en door het systeem in zijn totaliteit. Zo kan worden vastgelegd welk deel van de tijd de CPU bezet was, hoeveel Start-I/O instrukties (accessen op randapparaten) er per sekonde en per proces zijn gegeven, de gemiddelde grootte van de vrije ruimte, de gemiddelde lengte van de ready-to-run rij en de gemiddelde multiprogrammeringsgraad. Vooral deze algemene 'totaal'-informatie zal voor de prestatie-analyse van belang zijn.

Meestal kan de software-monitor zowel de accounting-gegevens als de meer algemene prestatie-gegevens verzamelen. Software-monitoren die door de leverancier bijgeleverd worden, zijn nogal eens geneigd resourcegebruik door het operatingsysteem niet in genoeg detail te vermelden en af en toe zelfs helemaal te veronachtzamen. Soms worden alleen maar de door de gebruikers expliciet aangevraagde systeemakties meegeteld. Het beslag dat het operatingsysteem op de resources legt, wordt meestal met *overhead* betiteld. Zulke monitoren specificeren deze 'overhead' niet voldoende.

Overhead —dat staat voor overlast— is een wat onvriendelijke naamgeving: overhead behoort geen last te zijn, het is werk dat in het belang van allen wordt uitgevoerd om —als het goed is— ieder te gerieven. Voor de accounting is het niet uitsplitsen van de overhead misschien verdedigbaar, de kosten van overhead moeten maar 'gezamenlijk' —hoe dan ook— gedragen worden. Maar voor een goede evaluatie van de prestaties kan het funest zijn als specifieke informatie over het gebruik van resources door het operatingsysteem ontbreekt.

Moderne monitoren zijn krachtig, ze bevatten natuurlijk steeds meer computergrafische mogelijkheden om overzichten in real time te laten zien en interaktief te bewerken. Nieuwe metingen kunnen à la minute worden opgestart en geanalyseerd. Discussies met verontruste gebruikers worden dan gevoerd aan de hand van gegevens over de situatie, zoals die zich op dat moment voordoet. Het is helemaal

volgens de 'state of the art' als de uitvoer van de monitor automatisch kan worden doorgegeven aan een op de machine draaiend analytisch model van de configuratie. Voor de grote mainframes zijn tegenwoordig zulke modellen beschikbaar. De gemeten karakteristieken funktioneren dan als invoer voor voorspellingen over het presteren onder alternatieve omstandigheden. Het computermodel levert deze 'stante pede'. Probleem bij deze faciliteit is nu nog dat de monitor soms systematisch een onvolledige informatie aanlevert (bijvoorbeeld wel alle user-SIO's, niet alle system-SIO's —een mankement van het soort dat we zojuist aangaven). De parameters uit het model moeten dan eerst aangepast (gecalibreerd) worden, totdat er een redelijke overeenstemming is tussen de door het model voorspelde huidige performance en de gemeten huidige performance. Verder is het nogal eens een 'fabrieksgeheim' van de makers van het model hoe bepaalde complexe zaken (bijvoorbeeld prioriteiten) in het model zijn ingebracht. Daardoor kunnen ook de uitkomsten van het model op die punten discutabel worden.

2.4. SYSTEEMBEHEER

De 'moderne' systeembeheerder kan profiteren van deze krachtige hulpmiddelen om performance te meten. Zijn taak is echter niet gering. De wisselwerking tussen het gedrag van de gebruikers en het gedrag van de machines is complex, de bepalende faktoren lopen uiteen van menselijke, 'psychologische', schijnbaar niet redelijke reakties, tot details van hardware-implementaties. De beheerder zal rekening moeten houden met eisen van zowel management als gebruikers. De leiding verlangt een optimaal gebruik van de beschikbare hulpmiddelen en een betrouwbare capaciteitsplanning voor de toekomst. De gebruikers hebben contractueel de verzekering gekregen dat ze per soort werk op een bepaalde service kunnen rekenen, en daar staan ze op. De beheerstaak zal daarom alleen goed uitgevoerd kunnen worden als er gerichte voorstellen kunnen worden overgelegd, die door waarnemingen zijn onderbouwd.

De systeembeheerders zullen de hulpmiddelen natuurlijk op de eerste plaats gebruiken voor het constateren van duidelijke mankementen: 'exception reporting', zoals 'die printer loopt onregelmatig, de transfertijd over die lijn is veel te hoog'. Het constateren van deze gebreken moet snel gebeuren, ook wie 'eerste hulp' verleent moet toegang hebben tot performance-tools.

Om te blijven voldoen aan de afgesproken service is het nodig in te spelen op de behoeften van de gebruikers. Veranderingen in het werkaanbod wijzen vaak op veranderingen in deze behoeften en het is zaak deze te constateren voordat ze expliciet geuit worden, omdat er tot die tijd nog preventief kan worden ingegrepen. Door een goede voorlichting kan het gebruikersgedrag worden bijgestuurd en wordt vermeden dat de beschikbare hulpmiddelen onhandig worden gebruikt (verkeerde blokkingsfactoren, compilers, enzovoort).

Uit de historische meetgegevens (historie is vaak al de dag van gisteren) kunnen door trend-analyse extrapolaties worden gedaan voor het komende gebruik van de resources en de aanstaande werklast. Zo kan een capaciteitsplanning ontstaan.

3

Inleiding operationele analyse

3.1. INLEIDING

Onze oriëntatie over de achtergronden van de prestatie-evaluatie komt nu bij het meer wiskundig geformuleerde deel: we worden geconfronteerd met de *operationele analyse*.

In de operationele analyse wordt geprobeerd om steeds rechtstreeks vanuit meetgegevens te redeneren, zonder veronderstellingen vooraf. In dit inleidende hoofdstuk wordt een begin gemaakt met deze aanpak. Er worden twee operationele schatters ingevoerd.

De operationele schatter voor de gemiddelde duur van een bepaalde toestand wordt gebruikt om informatie te krijgen over de 'echte' gemiddelde duur van deze toestand. Een operationele schatter is een voorschrift. Dit voorschrift luidt bij deze schatter:

> Ga enige tijd meten. Bepaal hoelang het systeem in die tijd in de bedoelde toestand is, die tijd noemen we *DUUR*. Bepaal ook hoe vaak het systeem in die tijd de toestand verlaat. Dat aantal noemen we *UIT*. De gemeten waarde van de operationele schatter voor de gemiddelde duur van de toestand is het quotiënt van *DUUR* en *UIT*.

De operationele schatter voor de kans op een bepaalde toestand wordt gebruikt om informatie te verkrijgen over de kansen op het aantreffen van toestanden. Het voorschrift bij deze schatter luidt:

> Ga enige tijd meten. Bepaal hoelang het systeem in die tijd in de bedoelde toestand is, die tijd noemen we *DUUR*. De meting duurt *MEETDUUR*. De gemeten waarde voor de operationele schatter voor de kans de toestand aan te treffen is het quotiënt van *DUUR* en *MEETDUUR*.

Het gebruik van deze voorschriften om informatie te krijgen over de gemiddelde duur van en de kans op een bepaalde toestand, ligt erg voor de hand. Iedereen zal in voorkomende gevallen *DUUR/UIT* uitrekenen en dat de gemeten gemiddelde toestandsduur noemen. En *DUUR/MEETDUUR* is voor elk de gemeten kans op de toestand. In dit hoofdstuk willen we deze intuïtieve rechtvaardiging vergelijken met preciezere overwegingen uit de statistiek en de stochastiek. Dan blijkt dat er toch wel wat vragen overblijven, de voornaamste is eigenlijk dat de voorgaande voorschriften veel te vaag zijn. Maar wie zich daar niet aan stoot zal dit hoofdstuk vooral gebruiken om de notaties en formuleringen te leren kennen en kan het als een 'intermezzo' zien.

3.2. TOESTANDEN

Een situatie wordt, heel neutraal, altijd beschreven als een bepaalde *toestand* van een *systeem*. Het begrip toestand is dus eigenlijk een synoniem van 'situatie'. Het systeem is in de computer-performance vaak het computersysteem of (onder)delen daarvan. In de loop van de tijd gaat het systeem van toestand naar toestand, want de situaties die zich bij het systeem voordoen veranderen voortdurend.

Een toestand moet worden gespecificeerd. Toestanden met dezelfde specificaties zijn dezelfde toestanden. Een specificatie kan uitgebreid en gedetailleerd zijn, maar ook heel ruim, zoals: er zijn drie terminalgebruikers (terminalisten) ingelogd. Het begrip toestand zal vooral worden gebruikt om aan te geven dat situaties op voor de vraagstelling relevante punten verschillen; als situaties op niet-relevante punten verschillen hebben, zullen het toch dezelfde toestanden zijn.

Er zal steeds 'iets' zijn dat verschillende waarden —of vormen, verschijningen— kan aannemen, de verschillende waarden bepalen de toestanden, ze dienen als specificatie van de toestand. Specificaties van toestanden zijn bijvoorbeeld: een aansluitpin heeft een positieve spanning (toestand 1) of staat op spanning 0 (toestand 0); van een buffer zijn n cellen gevuld: toestand n ($0 \leqslant n \leqslant$ buffergrootte); een processor is bezig met een zekere instruktie uit zijn repertoire; een randapparaat schrijft bepaalde blokken weg, terwijl de processor met andere blokken bezig is; de CPU heeft gedurende t msek. een programma uitgevoerd; de diskarm is over c cylinders verplaatst.

We zeggen steeds vrijblijvend, maar hopelijk concreet genoeg: er is een systeem dat in verschillende van elkaar te onderscheiden toestanden kan verkeren. Op ieder moment verkeert het systeem in precies één van deze toestanden. De toestandsspecificatie zal vaak vele, ook continu verlopende, waarden —parameters— bevatten. Het begrip toestand zullen we veel gebruiken, het brengt structuur aan.

3.3. METEN

Over een toestand wordt informatie verkregen door 'metingen' te verrichten. We zullen steeds direkt bij het begin van een probleemstelling aangeven welke metingen er precies gedaan worden. Als we vaststellen wat er met een systeem aan de hand is, spreken we altijd van *meten*. Meten is een heel algemeen begrip, ook tellen met streepjes en kruisjes of klokken met een stopwatch valt er onder; ieder verzamelen van gegevens is 'meten'. In de statistiek zouden we van 'waarnemen' spreken.

Bij computers worden metingen gedaan met monitoren. Die draaien een tijdlang mee, in die tijd worden een aantal waarnemingen van diverse grootheden verzameld. Deze verzameling bepaalt over welke typische grootheden uitspraken kunnen worden gedaan. Ze beperken het detail waarin een toestand gespecificeerd kan worden. Als de bezetting van een kanaal gemeten wordt zonder er op te letten wat de CPU op dat moment presteert, heeft het in eerste instantie geen zin over toestanden te spreken die gespecificeerd zijn als: CPU en kanaal beide bezet. Als er niet gelet wordt op het aantal aktieve gebruikers, zal men niet een toestand invoeren waarin drie aktieve gebruikers een bepaald aantal opdrachten per sekonde inbrengen.

Onbekende of minder relevante informatie wordt vaak gevangen in het vage begrip *heersende omstandigheden*. Er wordt dan gesproken over een zekere toestand 'onder de heersende omstandigheden'. Strikt genomen zou men de heersende omstandigheden in de specificatie van de toestand moeten opnemen. Het is echter gebruikelijk om dit niet te doen.

3.3.1. stochasten

Bij situaties in en rond een computerinstallatie hebben de grootheden die gemeten worden altijd een stochastisch karakter. Zulke grootheden hebben de volgende eigenschap: als er onder dezelfde omstandigheden wordt gemeten, komt er niet altijd hetzelfde uit. Meten in dezelfde toestand zal in het algemeen verschillende waarden opleveren. Deze grootheden worden beschreven met *stochasten*. De waarden van de stochast hebben een *spreiding* rond de 'echte' gemiddelde waarde. De 'echte' of 'werkelijke' gemiddelde waarde van de stochast wordt de *verwachtingswaarde E* genoemd. De spreiding van de stochast wordt gekarakteriseerd door zijn *variantie VAR*. De kansen op de mogelijke waarden van de stochast worden beschreven door zijn *verdeling*, eis is dat zo'n (kans)verdeling bestaat en 'fatsoenlijk' is. Uit de meetgegevens zal informatie volgen over de verdelingen van zulke stochasten. Zie voor het werken met verdelingen bijvoorbeeld §7.3.3 en §7.7.

3.4. METEN EN SCHATTEN

Van een situatie wordt altijd op de eerste plaats gemeten hoe lang de situatie achterelkaar optreedt. Of iets preciezer gezegd: hoelang een toestand zich onder de heersende omstandigheden voordoet.

We bekijken in het kort hoe hieruit informatie over de *'duren'* van de toestand wordt verkregen. De duur van een tijdinterval, waarin een bepaalde toestand achterelkaar ononderbroken onder de heersende omstandigheden optreedt —een toestandsduur of 'duur' van een toestand—, kunnen we aangeven met een stochast, tenminste als we aan de bepaling 'onder de heersende omstandigheden' een precieze betekenis kunnen geven. Uit de statistiek is bekend hoe er informatie over de verwachtingswaarde en de kansverdeling van zo'n stochast, zeg D (van Duur), kan worden verkregen: Meet n keer hoelang de toestand duurt. Statistische analyse maakt dan uitspraken mogelijk over de verwachtingswaarde $E(D)$ van de duur D van de toestand en zelfs over de verdeling van D. We kijken naar informatie over deze verwachtingswaarde $E(D)$.

De metingen kunnen worden verkregen door onder de heersende omstandigheden tijdens een interval van lengte *MEETDUUR*, de meetduur, te registreren wat er aan de hand is. Laat de toestand zich dan n keer hebben voorgedaan, dat wil zeggen de toestand werd n maal betreden en daarna ook weer verlaten; de toestand was n keer bezet. Het ligt voor de hand voor ieder (i) van die keren de tijdsduur D_i te meten vanaf het komen in de toestand tot aan het verlaten. De grootheid $°D$, gedefinieerd als

$$°D = \sum_{i=1}^{n} D_i / n$$

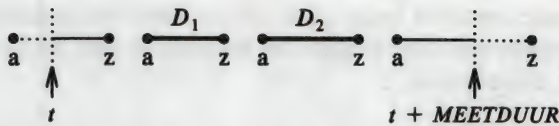
zal daarbij gebruikt worden als schatter voor de verwachtingswaarde $E(D)$ van de toestandsduur.

In veel meetsituaties is een variant op $°D$ aantrekkelijker. Er wordt alleen de totale tijd *DUUR* bijgehouden waarin tijdens de meetduur van duur *MEETDUUR* de toestand bezet is. Ook wordt geteld hoe vaak die bezet is, dit noemen we *UIT*. Een schatter voor $E(D)$ is dan

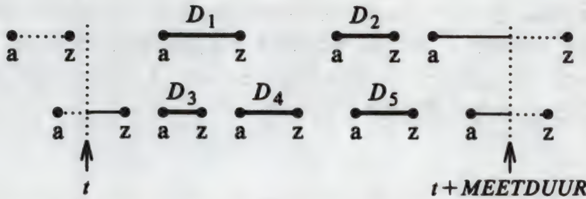
$$'D = DUUR / UIT$$

We noemen deze schatter $'D$ de *operationele schatter* voor de gemiddelde duur van een toestand. Operationele schatters geven we aan met een aanhalingsteken voor de grootheid. Bij de operationele schatter voor de gemiddelde duur worden de individuele duren niet bijgehouden!

De schatters $°D$ en $'D$ zijn verschillend. In het begin van de meting zit het systeem misschien al in de toestand en verlaat deze voor het eerst na een zekere tijdsduur. Tot dat moment zit het systeem dan in de toestand, maar die tijd telt niet mee als een D_i. Die tijd is ook geen deel van een D_i, maar draagt wel bij tot *DUUR*. Aan het eind van de meting zijn er dezelfde verschillen. Vergelijk de figuren 3.1-3.



Figuur 3.1. Systeem is telkens van a tot z in de toestand. Vet: bijdrage D_i. Getrokken: alleen bijdrage in *DUUR*. Gestippeld: geen bijdrage. De som van de vette plus de getrokken tijdsduren is *DUUR*. $n = 2$, $UIT = 3$.



Figuur 3.2. Parallel volgen van meerdere (hier twee) gelijkwaardige systemen. Vet: bijdrage D_i. Getrokken: alleen bijdrage in *DUUR*. Gestippeld: geen bijdrage. De som van de vette en de getrokken tijdsduren is *DUUR*. n = 5, *UIT* = 6.

De specificatie bij de operationele schatter van 'hoe vaak', dus van *UIT*, is een kwestie van afspraak. Het aantal malen bezet kan worden geteld als het aantal malen dat de toestand betreden wordt (keuze 1), of als het aantal malen dat de toestand verlaten wordt (keuze 2). Of misschien als het rekenkundig gemiddelde van deze twee (keuze 3). Deze aantallen verschillen. Neem bijvoorbeeld een systeem dat aan het begin van de meetduur in de toestand is en vervolgens binnen de meetduur de toestand (een of meer keer) verlaat. Als we voor *UIT* het aantal malen nemen dat de toestand betreden wordt (keuze 1) telt het eerste verlaten van de toestand niet mee in *UIT*. Wanneer we echter voor *UIT* het aantal malen nemen dat de toestand verlaten wordt (keuze 2) doet het eerste verlaten wel mee, net als bij keuze 3. We zullen kiezen voor keuze 2. Voortaan houden we ons aan de afspraak dat *UIT* het aantal malen telt dat de toestand binnen de meetduur verlaten wordt, zoals de naamgeving *UIT* suggereert.

De aangegeven verschillen tussen de beide schatters $°D$ en $'D$ zijn *randeffekten*. De verschilpunten zijn op het eerste gezicht, en als men niet precies is, van ondergeschikt belang. Voor behoorlijk grote waarden van n en *UIT* zullen ze relatief (meestal) weinig te betekenen hebben (vergelijk bijvoorbeeld §5.3, maar ook §11.3). Het zal blijken dat de operationele schatter $'D$ een meer rechtstreeks werkend hulpmiddel is (§4.5.2, §7.4). We zullen voortaan werken met de operationele schatter $'D$.

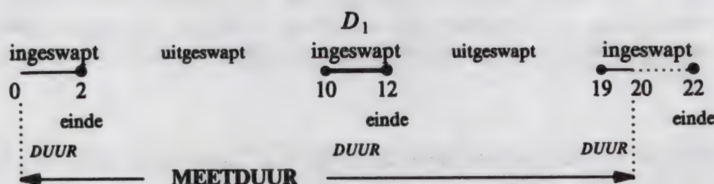
De meetgegevens *DUUR* en *UIT* zijn ook bruikbaar in nòg een operationele schatter. Tijdens de meetduur, die *MEETDUUR* duurt, zal tijdens *DUUR* tijdseenheden de betreffende toestand bezet zijn. De operationele schatter voor de kans dat onder de heersende omstandigheden het systeem op een willekeurig moment in de toestand wordt aangetroffen, is

$$'p_{toestand} = DUUR\,/\,MEETDUUR$$

Kort gezegd is dit de operationele schatter voor de kans op (het aantreffen van) de toestand.

3.4.1. voorbeeld: ingeswapt

Een proces kan pas door de CPU verwerkt worden als het is ingeswapt en daarmee 'in core' is. Op het tijdstip 0 is een bepaald programma in core, op tijdstip 2 wordt het uitgeswapt, op tijdstip 10 ingeswapt en op 12 uitgeswapt, op tijdstip 19 ingeswapt en op 22 uitgeswapt (figuur 3.3).



Figuur 3.3. Ingeswapt/uitgeswapt.

De relevante toestand is *'het programma is ingeswapt'*. Bij een meetduur beginnend bij tijdstip 0 en eindigend bij tijdstip 20 is de gemeten waarde voor de schatter $°D$ $(12-10)/1 = 2$. De gemeten waarde voor de schatter $'D$ is $((2-0)+(12-10)+(20-19))/2 = 5/2$, $UIT = 2$ omdat in *UIT* alleen het beëindigen van de toestand 'ingeswapt' op de tijdstippen 2 en 12 meetelt. De gemeten waarde voor $'p_{ingeswapt}$ is $5/20 = 1/4$.

3.5. SAMENVATTING

Deze eerste voorbeelden van operationele schatters zijn inleidend, we komen er later ook heel andere tegen van allerlei portee en vorm. Het juiste gebruik er van zal steeds besproken moeten worden. Maar we hebben tot nu toe nog niets onalledaags bericht; laten we het nog eens samenvatten.

Als 'gemeten' gemiddelde duur van een toestand wordt het quotiënt opgegeven van

- de totale tijd waarin de toestand bezet is, en
- het aantal malen dat de toestand bezet is.

Het recept, waarmee zo een geschatte waarde voor de 'echte' gemiddelde duur wordt bepaald, is het voorschrift van de 'operationele schatter' $'D$

$$'D = DUUR \,/\, UIT$$

waarin *DUUR* de totale tijd is dat tijdens de meetduur de toestand bezet is en *UIT* het aantal malen dat tijdens de meetduur de toestand verlaten wordt. De meting moet worden uitgevoerd *onder de heersende omstandigheden*.

Het recept voor het bepalen van een waarde van de operationele schatter $'p_{toestand}$ voor de kans het systeem in de toestand aan te treffen is

$$'p_{toestand} = DUUR \,/\, MEETDUUR$$

3.6. BETEKENIS OPERATIONELE SCHATTERS (MATHEMATISCH INTERMEZZO)

We moeten benadrukken dat beide schatters, $'D$ en $°D$, bedoeld zijn als heuse schatters in de gebruikelijke statistische zin. Het zijn dus op zich stochasten, het zijn grootheden waarvan pas het meten informatie binnen brengt. Een meting van *DUUR* en *UIT* in een bepaalde situatie levert één gemeten waarde voor de operationele schatter $'D$, zoals een meting van $\sum D_i \,/\, n$ één gemeten waarde geeft voor de schatter $°D$.

Het eigenlijke zwakke punt in deze inleiding zit helemaal in de vage toevoeging 'onder de heersende omstandigheden'. Deze begrenzing is niet verder toegelicht, er is bijvoorbeeld niet aangegeven hoe deze omstandigheden worden vastgelegd. Het is ook niet duidelijk of de duren D_i onderling onafhankelijk moeten worden verondersteld, ze kunnen zeker niet zonder meer worden gezien als lukrake 'trekkingen' uit een verdeling van D. Bij correlatie tussen de elkaar opvolgende waarden van D_i zullen de 'heersende omstandigheden' strikt genomen telkens anders zijn, een waarde hangt dan samen met de op hem volgende waarde. Op al dit soort punten zal verheldering moeten komen. Dit is nodig, alleen als aan 'onder de heersende omstandigheden' een statistisch zinvolle betekenis kan worden gegeven zijn $°D$ en de operationele schatter $'D$ schatters van $E(D)$ in de statistische zin.

De beide schatters zouden liefst zuivere schatters moeten zijn, hun verwachtingswaarde zou dan de waarde $E(D)$ zijn. Om dat aan te kunnen tonen zijn echter verdere specificaties nodig. Zo'n specificatie is bijvoorbeeld dat de toestanden zich ontwikkelen volgens een renewal-proces (§13.4). In een renewal-proces

zijn opeenvolgende duren onafhankelijk, het zijn daar inderdaad trekkingen uit dezelfde verdeling van de duur. Er kan dan worden verwezen naar de hoofdstelling van de renewal-theorie.

In de komende hoofdstukken zullen we verder ingaan op de elementaire aspekten van het meten aan systemen, die zich in de tijd ontwikkelen (bijvoorbeeld §7.8, §12.2). Nader commentaar op de operationele aanpak staat in §11.3.

3.7. OPGAVEN

opgave 3.1: tegenvoorbeeld?

De eenvoudige aanpak kan verkeerde suggesties wekken als het systeem absoluut niet voldoet aan de zo algemeen vaag beschreven omstandigheden uit §3.4. We bekijken een geval waarin er geen sprake is van een 'echte' gemiddelde toestandsduur.

Een bepaald systeem komt telkens in dezelfde toestand. Deze toestand duurt 1 of 2 tijdseenheden. Er is een vast patroon waarin deze duren op elkaar volgen. Na 2×3^i van lengte 1 volgen er 2×3^i van lengte 2. Daarna komen $2 \times 3^{i+1}$ van lengte 1, gevolgd door $2 \times 3^{i+1}$ van lengte 2. En zo gaat het voort ($i = 0, 1, 2, 3$ etc.). De eerste toestandsduur heeft lengte 1, de tweede heeft lengte 2 en daarna ontrolt zich het aangegeven patroon.

- Ga na dat de gemiddelde toestandsduren (vanaf de tweede toestandsduur) variëren tussen 1.25 en 1.5. Schets het patroon daarin.
- Ga na dat de limiet van de gemiddelde duur over de eerste n toestandsduren, voor het aantal duren n naar oneindig, niet bestaat.
- Bespreek welke informatie de schatters $°D$ en $'D$ hier opleveren; geef verschillen met de situatie uit §3.4.

4

De relatie van Little

4.1. INLEIDING

Grootheden die het presteren van een systeem (of van delen daarvan) karakteriseren, noemen we prestatiegrootheden. We zullen er vele leren kennen.

Tussen een aantal van hen bestaat een eenvoudige relatie, die erg nuttig is en vaak wordt toegepast. Deze luidt kortweg: gemiddeld aantal is gemiddelde duur maal gemiddelde stroom. De relatie wordt genoemd naar J.D.C. LITTLE, die de betrekking voor het eerst in alle exacte precisie bewees, het is de 'relatie van Little'. Het is een vrij voor de hand liggende betrekking, die waarschijnlijk ook door de niet-ingewijde op eigen gezag ('intuïtief') zou worden gehanteerd (maar hoe vaak?). We gaan nogal uitgebreid in op de relatie van Little, enerzijds omdat deze vaak zal worden gebruikt en de achtergrond van de relatie dus helemaal helder moet zijn, anderzijds om ervaring op te doen met de aanpak via operationele schatters.

4.2. METEN TERMINAL-SESSIES

4.2.1. gemiddeld aantal ingelogde terminals

Ons performance-probleem is het bepalen van de gemiddelde sessieduur en het gemiddelde aantal ingelogde terminals. Terminalisten zijn telkens slechts aktief tussen het moment van inloggen aan hun terminal en het eerstvolgende moment van uitloggen aan deze terminal. Zo'n periode is een *sessie*. Tijdens een bepaalde meetduur, die *MEETDUUR* duurt, worden gegevens bijgehouden over het in- en uitloggen van terminalisten. Hoe bepaalt een rapporteur hieruit het gemiddelde aantal ingelogde terminals?

Hij zal, misschien na enig gefilosofeer, overgaan tot het berekenen van de som van alle *sessieduren*. De som wordt gedeeld door de meetduur en het resultaat

genoteerd als het waargenomen gemiddelde aantal ingelogde terminals. Het is alsof de rapporteur een operationele schatter hanteert en de gemeten waarde daarvan vermeldt. Een gebruiker is tijdens een meetduur òf ingelogd òf niet ingelogd. Tijdens de meetduur wordt gemeten (de som is over de index *A* die alle gebruikers langs loopt die tijdens de meetduur ooit ingelogd zijn):

$$S_A = \textit{totale tijd, waarin terminalist } A \textit{ is ingelogd}$$

$$S = \textit{totale tijd, die ingelogd wordt doorgebracht} = \sum_A S_A$$

De gehanteerde schatter voor het gemiddelde aantal ingelogde terminals is

$$'s = S\,/\,MEETDUUR$$

De collectieve grootheid *S* is de som van alle binnen de meetduur vallende sessieduren, plus alle binnen de meetduur vallende parten van sessieduren die gedeeltelijk ook buiten de meetduur vallen. De laatste bijdragen vormen een randeffekt, ze preciseren wat meegenomen wordt (vergelijk de figuren 4.1 (§4.3.1) en 3.1-3).

De overwegingen om deze schatter te hanteren zijn waarschijnlijk de volgende. Het gaat om een collectief effekt, namelijk een gemiddeld aantal. Daarom moet een collectieve grootheid worden gevonden, waarin een steeds wisselend aantal gebruikers elk individueel bijdraagt. De 'individuele' inbreng van een terminalist moet groter zijn naarmate deze langer is ingelogd. Als er tijdens een tijdinterval *t* bijvoorbeeld drie terminalisten tegelijk zijn ingelogd moet hun gezamenlijke bijdrage over dit interval driemaal groter zijn dan wanneer er maar één enkele terminalist zou zijn ingelogd; verder is het gemiddelde aantal ingelogde terminals drie als *t* gelijk is aan de meetduur. De voor de hand liggende individuele bijdrage die aan al deze verlangens voldoet is de sessieduur. De collectieve grootheid is dan de som van de sessieduren. Als die door de meetduur wordt gedeeld ontstaat de gehanteerde schatter.

Laten we deze collectieve grootheid *S* nader bekijken. We kunnen elke collectieve grootheid altijd opgebouwd denken uit individuele bijdragen, zoals S_A, of uit collectieve 'multi'bijdragen. Zijn er bijvoorbeeld steeds drie terminalisten ingelogd tijdens het tijdinterval *t*, dan dragen deze in dit tijdinterval samen de multibijdrage $3t$ bij. Een ingelogde terminalist draagt tijdens zijn sessie één bij in het aantal ingelogde terminals, de individuele bijdrage door hem geleverd is eenmaal de sessieduur. Maar hij werkt ook een deel van zijn tijd (t_3) parallel met twee anderen en draagt dan bij in de multibijdrage $3t_3$, een ander deel van zijn tijd (t_4) zijn er vier terminalisten ingelogd en draagt hij bij in de multibijdrage $4t_4$ (enzovoort).

De individuele bijdragen worden parallel in de tijd geleverd, de multibijdragen worden collectief aangedragen. Het is erg nuttig om in de analyse van een systeem of subsysteem te switchen tussen een opbouw naar parallel geleverde bijdragen en

een opbouw naar collectieve bijdragen. Dit levert hier een nadere rechtvaardiging voor het hanteren van juist deze schatter:
Tijdens de meetduur *MEETDUUR* is in M_0 tijdseenheden geen enkele terminalist ingelogd, gedurende M_1 tijdseenheden 1 terminalist, gedurende M_i tijdseenheden i terminalisten, enzovoort. S wordt daarmee als volgt opgesplitst naar multibijdragen (de som loopt van 0 tot en met het hoogste aantal tijdens *MEETDUUR* tegelijk ingelogde terminalisten):

$$S = \sum_i i M_i$$

zodat

$$'s = \sum_i i M_i \,/\, MEETDUUR$$

en dat is

$$'s = \sum_i i \, 'p_i$$

We schreven, volgens §3.4 uit het hoofdstuk INLEIDING OPERATIONELE ANALYSE, $M_i \,/\, MEETDUUR$ als de operationele schatter $'p_i$ voor de kans op het aantreffen van de toestand *er zijn i terminals ingelogd*. De nu verkregen uitdrukking voor de schatter $'s$ voor het gemiddelde aantal ingelogde terminals is van de vorm *waarde×(kans op die waarde)*. Dit is de gewone statistische uitdrukking voor de verwachtingswaarde van een stochast, alleen is voor de kans de operationele uitdrukking ingevuld. Deze herschrijving zal de rapporteur er van overtuigen dat hij de correcte grootheid hanteert.

4.2.2. gemiddelde sessieduur, gemiddelde uitlogstroom

De rapporteur zoekt ook informatie over de gemiddelde sessieduur. Die vindt hij door de som van de sessieduren te delen door het aantal sessies, het resultaat noteert hij als de waargenomen gemiddelde sessieduur. We zetten ook dit over naar operationele schatters. Nu kunnen we aansluiten bij de discussie uit het hoofdstuk INLEIDING OPERATIONELE ANALYSE, want de sessieduur is de duur van een toestand. De betreffende toestand is *terminalist ingelogd*. Het aantal sessieduren wordt gepreciseerd als het aantal malen dat tijdens de meetduur een sessieduur wordt beëindigd door uit te loggen. Uit de meetgegevens over de meetduur

S = *totale tijd die ingelogd wordt doorgebracht*

$UITLOG$ = *aantal malen dat er een terminalist uitlogt*

volgt als operationele schatter voor de gemiddelde sessieduur

$$'S = S / UITLOG$$

Merk op dat in deze schatter bijdragen van mogelijk verschillende terminals(-isten) meedoen en dat bijdragen parallel in de tijd geleverd kunnen worden, omdat sessies gedeeltelijk parallel verlopen.

Het is van belang vast te stellen dat de operationele schatters voor het gemiddelde aantal ingelogde terminals en voor de gemiddelde sessieduur dezelfde collectieve grootheid bevatten, namelijk de totale inlogtijd S. Voor het gemiddelde aantal ingelogde terminals wordt S gedeeld door de meetduur, voor de gemiddelde sessieduur door het aantal terminalisten dat heeft uitgelogd. Dit verschil kan in verband worden gebracht met de uitlogstroom, die gedefinieerd wordt als het aantal terminalisten dat per tijdseenheid (t.e.) uitlogt. De uitlogstroom geeft dus de frequentie aan waarmee er wordt uitgelogd. Stroom en frequentie zijn equivalenten van het Engelse begrip 'rate', dat we ook vaak met snelheid verwoorden.

De operationele schatter voor de (gemiddelde) uitlogstroom is

$$'uitlogstroom = UITLOG / MEETDUUR$$

Uit de definities van de schatters volgt nu:

$$'s = S / MEETDUUR = (S / UITLOG)(UITLOG / MEETDUUR)$$

of

$$'s = 'S \times 'uitlogstroom$$

Voor de operationele schatters voor de gemiddelden geldt blijkbaar *gemiddeld aantal ingelogde terminals is gemiddelde sessieduur maal gemiddelde uitlogstroom.* Deze samenhang is een eerste voorbeeld van de *relatie van Little.*

4.3. RELATIE VAN LITTLE

Om concreet te zijn formuleerden we de vraagstelling voor sessies van terminalisten. De gevonden samenhang geldt natuurlijk veel algemener. Deze bestaat altijd in situaties waarin deelsystemen parallel in de tijd telkens eenzelfde toestand (hier: ingelogd) bezetten. Steeds is er dan een relatie tussen het gemiddelde aantal deelsystemen dat op een willekeurig moment in de toestand wordt aangetroffen (collectieve grootheid), de gemiddelde duur van het bezetten van de toestand door een deelsysteem (individuele grootheid) en het gemiddeld aantal keren dat per tijdseenheid de toestand verlaten wordt (collectieve grootheid). De gevonden

samenhang luidt algemeen, kort geformuleerd:

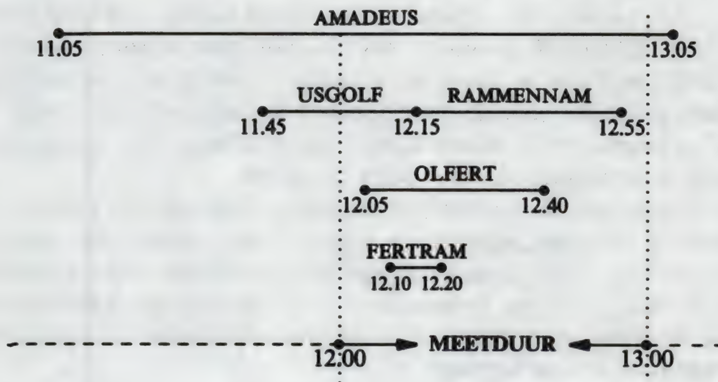
$$\textit{gemiddeld aantal} = (\textit{gemiddelde duur}) \times (\textit{gemiddelde stroom})$$

Zojuist verkregen we deze samenhang voor de operationele schatters voor het gemiddeld aantal ingelogde terminals, de gemiddelde sessieduur en de gemiddelde uitlogstroom.

Deze relatie is bekend uit de wachttijdentheorie (zie §4.5.1). In die theorie mag voor 'gemiddelde' de verwachtingswaarde worden gelezen, gemiddeld aantal is daar de verwachtingswaarde van het aantal bij de server (bediende) wachtenden of aanwezigen en gemiddelde duur is de gemiddelde wachtduur respectievelijk verblijfduur bij de server. De gevonden relatie is in de wachttijdentheorie een betrekking tussen verwachtingswaarden en heet de *Stelling van Little*. We zullen de kort geformuleerde vorm altijd betitelen als de *relatie van Little*; de betekenis van 'gemiddelde' moet uit de context blijken.

4.3.1. voorbeeld: inloggedrag

Amadeus logt 's morgens om 11.05 in en 's middags om 13.05 uit. Usgolf logt in om 11.45 en uit om 12.15. Olfert, Fertram en Rammennam loggen in om 12.05, 12.10 en 12.15; ze loggen uit om 12.40, 12.20 en 12.55.



Figuur 4.1. In- en uitloggen.

Er zijn 4 terminals beschikbaar, Rammennam gebruikt kennelijk de door Usgolf vrijgegeven terminal. Het verloop van hun aktiviteiten kan gemakkelijk worden gevolgd op 4 parallelle tijdassen, voor iedere terminal één (zie figuur 4.1).

Als wordt gemeten tussen 12.00 en 13.00 uur is (volgorde: Amadeus, Usgolf, Olfert, Fertram, Rammennam)

$$S = 60+15+35+10+40 = 160;\ UITLOG = 4$$

De gemeten waarden voor de operationele schatters zijn:

- *gemeten gemiddelde sessieduur* is 160/4 = 40 min.
- *gemeten gemiddeld aantal terminalisten* is 160/60 = 8/3 terminalist.
- *gemeten gemiddelde uitlogstroom* is 4/60 = 1/15 'logout' per minuut.

De relatie van Little is hier

$$8/3 = 40 \times 1/15$$

4.4. INTERPRETATIES VAN DE RELATIE VAN LITTLE

De samenhang werd gevonden als een verband tussen operationele schatters. Wat kan, algemeen gesproken, de betekenis zijn van zo'n verband?

Als een meting is verricht zijn de gemeten waarden van alle betrokken operationele schatters bekend. De gevonden samenhang levert voor deze gemeten waarden een identiteit op. Dit volgt rechtstreeks uit de definities van de operationele schatters, uit de manier waarop uit de metingen de gemeten waarden worden berekend. Zo worden bijvoorbeeld uit de meetwaarden van *MEETDUUR*, *S* en *UITLOG* de quotiënten *S/UITLOG*, *S/MEETDUUR* en *UITLOG/MEETDUUR* gevormd. Als die quotiënten worden ingevuld in de relatie van Little resulteert er exact een identiteit. Het is deze samenhang die we in §4.2 hebben gevonden voor de operationele schatters rond sessieduren.

Dit is het in zekere zin triviale aspekt van de relatie. Een meting levert zo geen bevestiging of verwerping van de relatie van Little.

De betrekking kan echter wel een belangrijk praktisch nut hebben als check op de consistentie van meetgegevens, die mogelijk met verschillende meetinstrumenten worden verkregen. Het gemiddelde aantal terminalisten bijvoorbeeld kan tijdens de meetduur ook gevonden worden door af en toe ('op lukrake momenten'; zie het hoofdstuk BEURTEN EN KANSEN) door de systeemsoftware te laten vastleggen hoeveel terminalisten er zijn ingelogd.

Deze beschrijving van de relatie van Little als een direkte samenhang tussen meetresultaten is altijd correct, we hebben in de vorige paragraaf eigenlijk de afleiding van deze *beperkte* vorm ('beperkte betekenis') van de relatie van Little gegeven.

Maar er is gewoonlijk een ruimere, verdergaande zingeving mogelijk. We introduceerden een operationele schatter als een schatter om een 'echte' gemiddelde waarde te schatten. Er is steeds een voorschrift waardoor deze schatter wordt aangegeven; de bedoelde echte waarde is de verwachtingswaarde van de

betrokken stochast. Zo kunnen de gemeten waarden voor de operationele schatter voor de gemiddelde sessieduur voeren tot een schatting in de vorm van een betrouwbaarheidsinterval voor de echte gemiddelde sessieduur, de verwachtingswaarde van de sessieduur. Daarom de volgende overweging:
Wanneer we het aantal metingen zouden verhogen —door vaker of langer waar te nemen— zouden de schattingen van alle schatters nauwkeuriger worden. Steeds zal daarbij de gevonden relatie van Little tussen de schatters geldig blijven. Het lijkt daarom niet gewaagd te veronderstellen dat dezelfde relatie ook moet bestaan tussen de grootheden die geschat worden.

Als deze gedachte juist is, komen we tot een belangrijke uitbreiding: de samenhang wordt een *verband tussen verwachtingswaarden* van op het systeem betrekking hebbende stochasten. De voor operationele schatters gevonden relaties zouden ook gelden voor de 'echte' gemiddelden! Voor elk systeem geldt dan 'volgens Little' dat het gemiddeld aantal gelijk is aan de gemiddelde duur maal de gemiddelde stroom, waarbij onder gemiddelde moet worden verstaan de betrokken verwachtingswaarde (bij de stroom is het handzaam te denken aan zijn inverse, de gemiddelde tussenpoos tussen de opeenvolgende momenten van vertrek uit de toestand; tussenpozen hebben gewoon een spreiding rond een gemiddelde).
Het is deze vorm die in de wachttijdentheorie door LITTLE voor een wachttijdensysteem werd bewezen: de STELLING VAN LITTLE.

We noemen dit de *uitgebreide* vorm van de relatie. De gevonden vorm met operationele schatters is de *beperkte* vorm. In de 'beperkte' vorm is de relatie van Little eigenlijk nog een stenen beeld, in de 'uitgebreide' vorm, als de *stelling van Little*, is hij tot leven gebracht.

Een precieze afleiding van de uitgebreide vorm is hiermee natuurlijk niet gegeven. Dat zou een nauwkeurige omschrijving vereisen van alle begrippen. Speciaal zou bij de operationele schatters een toevoeging in de geest van 'onder de heersende omstandigheden' naar voren komen en nauwgezet moeten worden gespecificeerd. De bewijsvoering kan dan bijvoorbeeld worden ingebed in de renewal-theorie (§13.4) en de theorie van regeneratieve processen, we gaan er niet verder op in.

Wel willen we wijzen op de allereerste hinderpaal die elke generalisatie van de beperkte vorm in de weg kan staan. Een operationele schatter als $'S$ (§4.2.2) of $'D$ (§3.4) (maar ook de schatter $^{\circ}D$) beslaat alleen dan een —liefst groot— aantal waarnemingen van een stochast als de *heersende omstandigheden* tijdens een meetduur gelijkwaardig blijven. Als die in de loop van de meetduur trendmatig veranderen wordt in de schatter ook nog 'over de omstandigheden' gemiddeld. We hebben dan niet informatie over vele metingen aan verschijningsvormen van één stochast, zeg X, maar over metingen aan ongelijkwaardige stochasten X_i, die realisaties voorstellen onder diverse omstandigheden, geïdentificeerd door i.

De kern van de kritiek op de 'operational analysis' richt zich op dit punt: zonder verdere aannamen is de generalisatie niet echt mogelijk en zal bij meting aan een systeem alleen de beperkte vorm betekenis hebben. Denning en Buzen stellen dat dit niet erg is, hun recept is om door verdere metingen aan hetzelfde systeem de eigenschappen van de heersende omstandigheden af te tasten en te vangen in begrippen als homogeniteit. Het zal inderdaad blijken dat het juist de kracht is

van de operationele zienswijze dat de daar gehanteerde gemiddelden ook *middelen over bijkomende nadere specificaties*, het zijn echt 'overall' gemiddelden.

In de wachttijdentheorie, een rijke theorie maar een 'theorie', zijn de omstandigheden goed gedefinieerd en blijkt de uitgebreide vorm van de relatie van Little (de Stelling dus) in de stationaire fase van het systeem een zeer wijd geldigheidsgebied te hebben. Het is nauwelijks een beperking aan te nemen dat het systeem of delen van het systeem zich als een wachttijdensysteem gedragen. De relatie van Little mogen we daarom ook in de uitgebreide vorm vrij onvervaard toepassen. En dat zullen we doen ook. De relatie/stelling van Little moeten we telkens weer gebruiken.

Uitgerekend in het gekozen voorbeeld zijn de omstandigheden vaak erg wisselend in de loop van de dag. Terminalisten zijn meestal blijvertjes. Waarschijnlijk is er geen wat langere periode aan te geven, waarover het patroon van inloggen stochastisch gezien essentieel hetzelfde blijft. Meerdere dagen —dus meerdere keren— meten over wat kortere perioden moet dan de oplossing brengen. Wat jammer voor het voorbeeld? Maar de relatie van Little blijft overeind staan.

Deze eerste beschrijving van een samenhang tussen operationele inzichten en stochastisch/statistische overwegingen moet wel wat vaag zijn gebleven. In de volgende hoofdstukken zullen telkens bij redeneringen met stochastische grootheden conclusies opduiken, die op operationele schatters gebaseerd kunnen worden. De verbanden krijgen daarbij gaandeweg meer contouren.

4.5. VORMEN VAN DE RELATIE VAN LITTLE

De relatie van Little, die we zo hebben leren kennen, zegt heel algemeen dat wanneer parallel in de tijd soortgelijke toestanden optreden, altijd geldt:

gemiddeld aantal = (gemiddelde duur) × (gemiddelde stroom)

Hierin is

- gemiddeld aantal = gemiddeld aantal dat op een lukraak moment in zo'n toestand wordt aangetroffen.
- gemiddelde duur = gemiddelde duur van het bezetten van zo'n toestand.
- gemiddelde stroom = gemiddeld aantal keren dat al deze toestanden per tijdseenheid verlaten worden.

Voor 'gemiddelde' moet òf —in de uitgebreide vorm— steeds verwachtingswaarde worden gelezen òf —in de beperkte vorm— steeds gemeten gemiddelde.

4.5.1. Little in de wachttijdentheorie

De relatie van Little wordt meestal zonder meer verbonden met de wachttijdentheorie. Daarin treedt de relatie op als de stelling '$L = \lambda W$'. In de stationaire fase van een wachttijdensysteem geldt

$$L = \lambda W$$

met L is het gemiddelde aantal wachtende klanten in het systeem, λ het aantal klanten dat het systeem per tijdseenheid binnengaat en W is de gemiddelde tijd (wachtduur) die een klant in het systeem wachtend doorbrengt (in een notatie die bij wachttijdensystemen veel wordt gevonden). De relatie geldt los van de optredende verdelingen, afhankelijkheden en afwikkelingsregelingen, het is de Stelling van Little.

De toestand *wachtend* wordt parallel in de tijd bezet door een wisselend aantal klanten, L is het gemiddeld aantal in deze toestand. W is de gemiddelde duur van het bezetten van de toestand *wachtend*. In de stationaire fase is het gemiddeld aantal klanten dat per tijdseenheid binnentreedt, gelijk aan het gemiddeld aantal dat per tijdseenheid vertrekt. De grootheid λ correspondeert dan ook met de gemiddelde stroom (en wel met de doorstroom door het wachttijdensysteem).

De relatie (stelling) van Little verbindt op dezelfde manier ook het gemiddeld aantal klanten in het wachttijdensysteem, de gemiddelde verblijfduur in het systeem en λ. De relevante toestand is dan *wachtend of onder behandeling*. In §14.7.2 herhalen we de relaties van Little voor de wachttijdentheorie.

4.5.2. vervolg voorbeeld: inloggedrag (4.3.1); alternatieven voor afspraken operationele schatters

Uit de gemeten sessieduren van Amadeus c.s. volgde voor de gemeten waarden van de operationele schatters $8/3 = 40 \times 1/15$. Dit is de beperkte vorm van de relatie van Little, toegepast op de gemiddelden van inlogduur, aantal ingelogde terminalisten en uitlogstroom in de meetperiode. Uiteraard is $8/3 = 40 \times 1/15$: de relatie van Little is in de 'beperkte' vorm een identiteit.

Wanneer we, als in $°D$ uit §3.4, alleen de volledig waargenomen sessies meenemen, tellen alleen de sessies van Olfert, Fertram en Rammennam. De gemeten gemiddelde sessieduur wordt dan $(35+10+40)/3 = 28\,1/3$ (vergelijk figuur 4.1). Maar in deze vorm geldt niet $8/3 = 28\,1/3 \times 1/15$.

Als we in *UITLOG* de aantallen keren tellen dat is ingelogd (de login's en niet de logout's), wordt *UITLOG* = 3 en de gemeten gemiddelde sessieduur $160/3 = 53\,1/3$.

Zulke sterk verschillende uitkomsten —40, 28 1/3, 53 1/3— bij een simpel verschil in afspraak, ontstaan doordat het aantal waarnemingen te beperkt is, zodat randeffekten een grote rol spelen; welke waarde zal worden gerapporteerd wordt voornamelijk een kwestie van smaak.

Van de fanatieke Amadeus worden noch het uitloggen, noch het inloggen in de schatters meegeteld. Als de anderen niet hadden ingelogd, was *UITLOG* = 0 geweest en de gemeten waarde voor de sessieduur 60/0 minuten, dus onzinnig. Doordat het meetinterval van 1 uur voor Amadeus' gedrag kennelijk te kort is, valt trouwens op geen enkele manier een zinnige waarde te bedenken. In een 'robuust' programma dat meetresultaten automatisch verwerkt (in een software-monitor) moet natuurlijk altijd een regel worden ingebouwd om zulke extreme waarden probleemloos te verwerken.

Verreikende uitspraken zijn met dit beperkte waarnemingsmateriaal uiteraard niet te doen. Kennelijk veranderen de omstandigheden rond 12.05 uur, omdat dan een frisse groep zich meldt. Denk eens in dat van 11.18 tot 12.18 gemeten werd.

4.6. TOEPASSINGEN

4.6.1. voorbeeld: ziekenhuisbedden

Hoe alledaags de relatie van Little wel niet is, laat het volgende voorbeeld zien. In een academisch ziekenhuis werden volgens een Ziekenhuis Informatie Systeem in een jaar 16 000 patiënten opgenomen, het aantal geleverde verpleegdagen bedroeg 297 000. Het jaar telde 366 dagen. Rapportage: het gemiddeld aantal bezette bedden bedroeg 297000/366 bedden, de gemiddelde duur van een opname (ligduur) was 297000/16000 dagen, de gemiddelde opnamefrequentie was 16000/366 patiënten per dag. Deze gemiddelde cijfers zijn even 'gemiddeld' als die bij de rapportage over de dienstverlening door een computersysteem!

Nu de minister van volksgezondheid van de ziekenhuisdirekties een steeds kortere ligduur eist, omdat de gezondheidszorg veel te duur wordt, wordt er gereageerd met opvoeren van de opnamefrequentie —geheel volgens Little: de bezettingsgraad van het beddenbestand blijft op peil.

Ook in de discussie rond het cellentekort in gevangenissen treedt de relatie van Little op. In West-Duitsland worden per hoofd van de bevolking per jaar minder mensen tot gevangenisstraf veroordeeld dan in Nederland. Per hoofd van de bevolking zitten in de Bondsrepubliek meer mensen in de gevangenis dan in Nederland. De gemiddelde duur van de vrijheidsstraf is in Nederland dus korter.

4.6.2. voorbeeld: multiprogrammeringsgraad en gemiddelde verblijfduur

We passen de relatie van Little toe op het centrale deel van een computerconfiguratie.

De multiprogrammeringsgraad is het aantal opdrachten (programma's) dat tegelijkertijd is toegelaten tot het centrale deel van het computersysteem. De doorstroom van een computersysteem is het aantal per tijdseenheid verwerkte opdrachten (zie bijvoorbeeld §5.2). Bij een doorstroom van 10 opdrachten per sekonde en een gemiddelde multiprogrammeringsgraad van 3.2 opdrachten is de gemiddelde verblijfduur van een opdracht in het centrale deel van het computersysteem 'volgens Little' 3.2/10 = 0.32 sekonden. Deze conclusie geldt altijd,

hoe ingewikkeld het operatingsysteem ook funktioneert en hoe divers de opdrachten ook zijn: de relatie van Little is een fundamentele relatie!

De relatie van Little kan vervolgens worden toegepast op het computersysteem als geheel. Bij een gemiddelde responsetijd van 2 sekonden zegt Little ook dat er gemiddeld $10 \times 2 = 20$ opdrachten in de responsefase zijn, dus in het systeem verblijven. Er wachten dus gemiddeld $20 - 3.2 = 16.8$ opdrachten op toelating tot de verzameling onder multiprogrammering draaiende opdrachten. Deze opdrachten verblijven op 'spool areas', zijn 'uitgeswapt', enzovoort. Een opdracht zal gemiddeld $2 - 0.32 = 1.68$ sekonden op toelating wachten.

En de relatie kan toegepast worden op de situatie bij elk device apart! We brengen nog in dat de gemiddelde CPU-tijd per opdracht 60 msek. is; de CPU-tijd is de som van de tijdsduren waarin de CPU aktief bezig is met de opdracht: de som van de tijd in user mode en in privileged mode (system mode). Nu geldt verder volgens Little dat er gemiddeld $10 \times 0.060 = 0.6$ opdrachten door de CPU verwerkt worden. De bezettingsgraad (U) van de CPU zal dus 60% zijn. Er zijn dan gemiddeld $3.2 - 0.6 = 2.6$ opdrachten die door de randapparaten verwerkt worden, daar wachten op verwerking of wachten op verwerking door de CPU. Opdrachten zijn gemiddeld gedurende $0.32 - 0.06 = 0.26$ sekonden in een dergelijke situatie.

Het toepassen van de relatie van Little op grotere en kleinere deelsystemen van de configuratie geeft dus veel globale, maar relevante informatie. Merk op dat de doorstroom aan opdrachten voor ieder deelsysteem 'per definitie' dezelfde is, hier 10 opdrachten per sekonde (vergelijk §9.9.3). De getoonde relaties zullen bij het ontwikkelen van algoritmen voor het doorrekenen van computermodellen grote diensten bewijzen.

4.7. SAMENVATTING

Prestatiegrootheden worden gekarakteriseerd door aan te geven hoe ze gemeten worden. Er blijkt tussen de gemeten waarden voor gemiddeld aantal, duur en stroom een identiteit te bestaan. De identiteit kan worden gegeneraliseerd tot de 'alom geldige' relatie van Little. Voor de betreffende meetresultaten —de gemeten waarden van de operationele schatters— geldt altijd: als je twee van de drie gemiddelden hebt, is de derde ook bekend. Little zegt dat dit voor de 'echte' gemiddelden ook zo is. De gemiddelden van aantal, duur en stroom zijn dus ten nauwste verbonden. Als de waarden voor twee bekend zijn volgt de derde door vermenigvuldiging of deling.

4.8. OPGAVEN

opgave 4.1: bezetting terminals

Er zijn 6 terminals op een computersysteem aangesloten. Af en toe logt iemand in op een van deze terminals, tenminste wanneer het systeembeheer op dat moment de terminals voor hem als aangesloten heeft gedeclareerd. Aan de terminals is

gemeten gedurende een meetduur van 3000 t.e. (tijdseenheden). We geven het begin van deze meetduur aan als (tijdstip) 0 en het eind als (tijdstip) 3000. De terminals 1 en 2 zijn aangesloten van 0 tot 1000 en van 1500 tot 3000. De terminals 3, 4 en 5 zijn de gehele meetduur aangesloten. Terminal 6 is slechts van 1000 tot 2500 aangesloten.

Zowel in het tijdinterval 0-1000 als in het tijdinterval 1000-3000 zijn de terminals 3, 4 en 5 de helft van de tijd bezet geweest. Terminals 1 en 2 waren voor driekwart van de aangesloten tijd bezet: er was iemand ingelogd. Terminal 6 was, toen hij was aangesloten, voor de helft van de tijd bezet. In het tijdinterval 0-1000 hebben op elk van de terminals 1-5 50 terminalisten uitgelogd; in het tijdinterval 1000-3000 hebben aan elk van de terminals 1-5 100 terminalisten gewerkt (100 logouts) en 50 aan terminal 6.

Over het interval 0-1000 kunnen nu het gemiddelde aantal ingelogde terminals, het gemiddelde aantal aangesloten terminals, de gemiddelde inlogduur en de gemiddelde uitlogstroom worden berekend. Idem voor het interval 1000-3000. Maar ook voor het totale interval 0-3000 zijn deze grootheden volgens de operationele schatters te bepalen. Steeds wordt voldaan aan de relatie van Little en wel in de beperkte vorm voor de gemeten waarden van de operationele schatters.

Er kan aan de hand van deze operationele schatters direkt worden nagegaan met welke gewichten de gemiddelde waarden over de intervallen 0-1000 en 1000-3000 optreden in de overall gemiddelde waarden over het interval 0-3000.

Deze opgave komen we, in iets gewijzigde vorm, nogmaals tegen in het hoofdstuk RESPONSETIJDRELATIES, zie §5.7 (daar is dus de numerieke uitwerking te vinden).

5

Responsetijdrelaties

5.1. INLEIDING

De responsetijdrelaties vormen een basis voor iedere discussie over het verband tussen het aantal terminals, de denktijd aan die terminals en de door de terminalisten aan die terminals verkregen response van het computersysteem. Bij interaktieve verwerking zal voor elke 'soort' werk en voor elke 'soort' gebruiker, *apart* of *samengenomen*, steeds gelden:

$$\textit{gemiddelde responsetijd} = (\textit{gemiddeld aantal gebruikers})/\textit{doorstroom} - \textit{gemiddelde denktijd}$$

Zo'n responsetijdrelatie is eigenlijk niets anders dan een toepassing van de relatie van Little; elke responsetijdrelatie kan altijd via 'Little' gevonden worden. Toch zullen we de relaties vanuit de operationele schatters introduceren. We gaan wat nader in op de middeling, die in operationele schatters ligt opgesloten; het waarnemen wordt bewust eenvoudig gehouden, wat maakt dat zelfs de 'beperkte' vorm van een responsetijdrelatie niet strikt als identiteit aan de relatie voldoet.

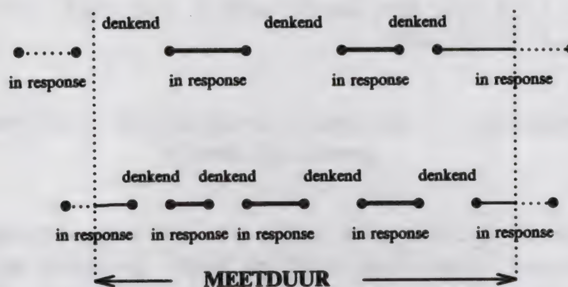
5.2. METEN INTERAKTIEVE VERWERKING

Een computersysteem verwerkt opdrachten die door gebruikers voor verwerking worden aangeboden. De gebruikers brengen hun opdrachten in via terminals dan wel via de batch. Zodra de gebruiker een opdracht heeft ingebracht gaat hij gedurende de *responsetijd* wachten op de verwerking van de opdracht. Nadat zijn opdracht is verwerkt, denkt hij gedurende de *denktijd* voordat hij de volgende opdracht inbrengt.

Iedere gebruiker brengt zijn tijd dus afwisselend wachtend op response en denkend door (figuur 5.1). Op een willekeurig moment waargenomen heeft hij òf een opdracht voor bewerking ingebracht en is in de toestand *in response* òf geen opdracht ingebracht en is in de toestand *denkend*. De identiteit van een gebruiker is niet relevant, meerdere individuen kunnen elkaar opvolgen als eenzelfde gebruiker.

We ontlenen de terminologie aan verwerking via terminals. De ingebrachte verzoeken voor verwerking heten opdrachten, het zijn commando's, jobs, programma's, enzovoort. Voor de batchgebruikers is het gebruikelijk de tijd waarin de opdracht is ingebracht, maar nog niet uitgeleverd, de *doorlooptijd* of *turnaround-tijd* te noemen; er is daar geen naam ingeburgerd voor de tijdsduur waarin de batchgebruiker niets aanlevert. Responsetijd en doorlooptijd zijn dus dezelfde begrippen.

De betekenisvolle naamgeving denktijd kan verwarrend zijn. Heel wat denkwerk wordt gedaan in de tijd waarin op response wordt gewacht, zodat de denktijd vaak niet meer is dan een reaktietijd, nodig voor het vormen van de volgende opdracht. Natuurlijk bepaalt uiteindelijk het meetinstrument wat er precies als responsetijd zal gelden en wat als denktijd wordt meegenomen.



Figuur 5.1. Parallel volgen van twee gebruikers. $IN = 7$, $UIT = 6$.

We veronderstellen dat er s gebruikers bezig zijn in de meetduur van duur *MEETDUUR*. Een bepaalde gebruiker fluktueert tussen de toestanden 'in response' en 'denkend'. Voor elk van deze toestanden worden gegevens bijgehouden. Voor gebruiker A zal in de meetduur worden gevonden, dat IN_A maal een denktijd beëindigd werd en door hem een volgende opdracht ingebracht, terwijl UIT_A maal een responsetijd beëindigd werd doordat een door hem ingebrachte opdracht bleek afgewerkt. Gebruiker A vertoefde in totaal gedurende R_A tijdseenheden in de toestand 'in response' en gedurende Z_A tijdseenheden in de toestand 'denkend'.

Deze meetgegevens behelsen:

$$IN_A = \text{aantal door gebruiker A ingebrachte opdrachten.}$$
$$UIT_A = \text{aantal verwerkte opdrachten, door A ingebracht.}$$
$$R_A = \text{totale tijd waarin gebruiker A wacht op response.}$$
$$Z_A = \text{totale tijd waarin gebruiker A denkt.}$$

De operationele schatter voor de *gemiddelde responsetijd van gebruiker* A is volgens de afspraak uit het hoofdstuk INLEIDING OPERATIONELE SCHATTERS (§3.4)

$$'R_A = R_A \,/\, UIT_A$$

En de operationele schatter voor zijn *gemiddelde denktijd* is volgens dezelfde afspraak, IN_A is ook het aantal beëindigingen van de toestand 'denkend',

$$'Z_A = Z_A \,/\, IN_A$$

Voor de *s* gebruikers samen leveren de metingen collectieve meetgegevens over de meetduur *MEETDUUR* (de som is over de index *A* die alle *s* gebruikers langs gaat):

$$IN = \text{aantal ingebrachte opdrachten} = \sum_A IN_A$$
$$UIT = \text{aantal verwerkte opdrachten} = \sum_A UIT_A$$
$$R = \text{totale tijd waarin op response werd gewacht} = \sum_A R_A$$
$$Z = \text{totale tijd waarin werd gedacht} = \sum_A Z_A$$

Merk op dat *R* geen tijdinterval is, ook geen 'niet aaneengesloten tijdinterval' zoals R_A, maar een som van vaak parallelle bijdragen. Zijn de gebruikers A en B tegelijk in de responsefase tijdens een bepaald tijdinterval van lengte *t*, dan dragen ze in die tijd samen $2t$ bij tot de collectieve grootheid *R*.

De grootheid *R* omvat vele realisaties van een responsetijd, afkomstig van alle gebruikers met elkaar. De overwegingen, die tot de operationele schatter $'D$ in §3.4 leidden, maken ook dat we het quotiënt van *R* en *UIT* gaan zien als de operationele schatter van een gemiddelde responsetijd. Het betreft natuurlijk de gemiddelde responsetijd, genomen over alle gebruikers. We zullen deze ter onderscheiding van de gemiddelde responsetijd van een individuele gebruiker vaak de *overall* gemiddelde responsetijd noemen.

De operationele schatter voor de 'overall' gemiddelde responsetijd is

$$'R = R\,/\,UIT \qquad \text{(dimensie: t.e.)}$$

en de operationele schatter voor de *overall gemiddelde denktijd* is

$$'Z = Z\,/\,IN \qquad \text{(dimensie: t.e.)}$$

Het *gemiddeld aantal per tijdseenheid verwerkte opdrachten* is een erg belangrijke en veel benutte performance-grootheid. Deze grootheid heet in het Engels throughput en in het Duits Durchsatz, wij zullen er het Nederlandse woord *doorstroom* voor gebruiken. Stroom wijst op het aspekt snelheid, aangegeven door de factor $tijd^{-1}$ in de dimensie. In de betiteling doorvoer of doorzet, die tegenwoordig nogal eens als substituut voor throughput wordt gebruikt, zou dit minder duidelijk tot uitdrukking komen.

De operationele schatter voor de (overall (gemiddelde)) *doorstroom* is

$$'doorstroom = UIT\,/\,MEETDUUR \qquad \text{(dimensie: t.e.}^{-1}\text{)}$$

De middeling is hier over de (stochastische en wellicht systematische) fluktuaties in de doorstroom, die zich tijdens de meetduur voordoen. In het begrip doorstroom gaat het —per definitie— om een gemiddeld aantal per tijdseenheid; er is dus zonder meer een middeling over stationaire statistische fluktuaties in het tijdpatroon inbegrepen. Alleen wanneer we willen benadrukken† dat er daarnaast ook over de in de tijd veranderende omstandigheden, dus over meer systematische fluktuaties, wordt gemiddeld, spreken we van *gemiddelde doorstroom* en noemen we de schatter de operationele schatter voor de *(overall) gemiddelde doorstroom*.

5.3. RESPONSETIJDRELATIES

De achtergrond van alle responsetijdrelaties kan nu aangegeven worden: de operationele schatters van de gemiddelde responsetijd, doorstroom en denktijd zijn niet onafhankelijk, maar hangen eenvoudig samen.

Dit volgt rechtstreeks uit hun definities. Door de afwisseling van responsetijd en denktijd maken bij iedere gebruiker A de totale responsetijd R_A en de totale denktijd Z_A samen de meetduur *MEETDUUR* vol:

$$R_A + Z_A = MEETDUUR$$

---

† Bij gemiddelde stroom uit de relatie van Little speelt hetzelfde.

Uit de definities van de collectieve grootheden R en Z volgt dan met s gebruikers parallel

$$R + Z = s \times MEETDUUR$$

Herschrijven van de schatters voor de gemiddelde (overall) responsetijd levert hiermee

$$\begin{aligned} 'R &= R / UIT \\ &= s \times (MEETDUUR / UIT) - Z / UIT \\ &= s / (UIT / MEETDUUR) - (IN / UIT)\,(Z / IN) \end{aligned}$$

of

$$'R = s / '\mathit{doorstroom} - (IN / UIT)\,'Z$$

Dit is het eenvoudige verband tussen de operationele schatters voor de overall gemiddelde responsetijd, denktijd en doorstroom.

Omdat wachten op response en denken elkaar afwisselen kan bij gebruiker A het verschil tussen IN_A en UIT_A absoluut genomen nooit groter worden dan 1. Het verschil tussen IN en UIT is absoluut genomen nooit groter dan s (zie de figuren 3.1-3 en 5.1). Als de meetduur $MEETDUUR$ lang genoeg is, beter gezegd als IN en UIT groot genoeg worden, zal de verhouding IN / UIT vrijwel 1 zijn en kan men rustig schrijven

$$'R = s / '\mathit{doorstroom} - 'Z$$

We noemen de beide relaties —zowel met als zonder IN / UIT— de overall *responsetijdrelatie,* een relatie voor alle gebruikers samen. Deze relatie, ofwel

$$'R + 'Z = s / '\mathit{doorstroom}$$

zegt hier dus dat voor de operationele schatters van de overall gemiddelden geldt dat de som van de gemiddelde responsetijd en de gemiddelde denktijd gelijk is aan het aantal gebruikers, gedeeld door de (gemiddelde) doorstroom.

In de operationele schatters is een middeling begrepen over alle in beschouwing genomen gebruikers; alle gebruikers zijn over één kam geschoren. Zo'n middeling is eigenlijk altijd aanwezig in operationele schatters. Het samennemen tot

$$'R = R / UIT \quad = \quad \sum_{A=1}^{s} R_A \Big/ \sum_{A=1}^{s} UIT_A$$

is echter wel op een verstandige manier gebeurd. In sommige commerciële rapportages en accounting-pakketten wordt als overall gemiddelde het gemiddelde van de gemiddelden per terminal genomen; men stelt (de som loopt over alle s terminals)

$$\text{overall gemiddelde responsetijd} = \left(\sum_{A=1}^{s} R_A \,/\, UIT_A \right) / \, s$$

Dit is statistisch onhoudbaar. De gemiddelden per terminal kunnen op heel verschillende aantallen responsetijden slaan en daardoor uiteenlopen in (statistische) nauwkeurigheid.

Ook voor iedere gebruiker apart bestaat er een responsetijdrelatie. De operationele schatter voor de (gemiddelde) *doorstroom aan opdrachten van gebruiker A* is

$$'doorstroom_A = UIT_A \,/\, MEETDUUR$$

Uitschrijven van R_A levert als *responsetijdrelatie voor gebruiker A:*

$$'R_A = 1 \,/\, 'doorstroom_A - (IN_A \,/\, UIT_A) \; 'Z_A$$

Merk op dat het verschil van *UIT* en *IN* (of UIT_A en IN_A) weer een randeffekt is (§3.4). Meestal zal men voor een eenvoudige vorm van waarnemen kiezen en dus òf *IN* òf *UIT* tellen, voor grotere aantallen waarnemingen —en die heeft men nodig— komt dat toch 'vrijwel' op hetzelfde neer. Het verwaarlozen van het verschil tussen *IN* en *UIT* is dus het introduceren van een meetfout, die men in de hand heeft. Het is geen statistische fout, niet het weglaten van een betrouwbaarheidsinterval in zijn verantwoording. Dit kan gezegd worden bij elk manipuleren van eindige randeffekten.

De aanpak volgens de operationele schatters lijkt op die bij simulatie. Ieder waarnemen (meten) aan een systeem levert één meetwaarde op voor de relevante operationele schatter. Er wordt vastgesteld welk pad door de tijd door het systeem is genomen, over dat pad worden de grootheden waarin men geïnteresseerd is, bijgehouden. Een simulatierun speelt precies één zo'n pad na, en wel voor alleen de grootheden waarin men belang stelt. Ook bij simulatie moet men het hoofd buigen over de vraag hoe men de run afbreekt: wat dragen randeffekten bij.

5.3.1. voorbeeld: denk/responsetijd

Er zijn 10 terminals aangesloten op een computersysteem. Een persoon achter een terminal (een terminalist) werkt in principe als volgt: geef opdracht een ingetoetste opdracht te verwerken, wacht gedurende een bepaalde tijd (responsetijd) op de melding dat deze opdracht verwerkt is, denk een poosje (denktijd) na, geef de volgende opdracht, enzovoort. Gedurende een representatieve tijdsduur van 1000

sekonden worden metingen verricht aan dit systeem. Tijdens deze meettijd zijn voortdurend alle 10 terminals bezet. Men vindt het volgende:

> in die tijd zijn er 400 opdrachten voor verwerking aangeboden en er zijn 402 opdrachten verwerkt gemeld.
>
> de terminalisten aan de terminals 1 en 2 hebben elk gedurende 200 sekonden gedacht, de terminalisten aan de terminals 3 tot en met 10 hebben elk gedurende 250 sekonden gedacht.

De meetwaarden zijn (in sekonden) $Z_1 = Z_2 = 200$, Z_3 tot en met $Z_{10} = 250$, $s = 10$. Aan elke terminal is 1000 sekonden gewerkt, dus ook $R_1 = R_2 = 800$, R_3 tot en met $R_{10} = 750$. Verder is $IN = 400$ en $UIT = 402$; de separate IN_1, UIT_1, IN_2, UIT_2 etc. zijn niet bijgehouden.

IN	$= 400$ opdr.
UIT	$= 402$ opdr.
$Z_1 = Z_2$	$= 200$ sek.
Z_3 t/m Z_{10}	$= 250$ sek.
$R_1 = R_2$	$= 800$ sek.
R_3 t/m R_{10}	$= 750$ sek.
s	$= 10$ term.

Tabel 5.1. Meetwaarden denk/responsetijd.

De gemeten waarde van $'R$ en $'Z$ zijn

$$'R = (2\times800+8\times750)/402 = 7600/402 = 18.9 \text{ sek.}$$
$$'Z = (2\times200+8\times250)/400 = 2400/400 = 6.0 \text{ sek.}$$

De gemeten waarde voor de operationele schatter voor de doorstroom is

$$'doorstroom = 402/1000 = 0.402 \text{ opdrachten per sekonde}$$

De responsetijdrelatie houdt hier in

$$7600/402 = 10/0.402 - 400/402\times6$$

De gemeten gemiddelde overall denktijd is 6.0 sekonden, de gemeten gemiddelde overall responsetijd is vrijwel 7600/400 = 19.0 sekonden, de gemeten doorstroom is

vrijwel 0.400 opdrachten per sekonde. Met ook als responsetijdrelatie

$$19.0 = 10/0.400 - 6.0$$

5.4. BETEKENIS VAN DE RESPONSETIJDRELATIES

We vonden de responsetijdrelaties als betrekkingen tussen operationele schatters. Het zijn dus voorbeelden van tussen operationele grootheden bestaande 'algemene relaties', waarvan we de betekenis bij de relatie van Little bespraken (§4.4).

De *'beperkte'* vorm, die we nu gevonden hebben, kan dienen om berekeningen te controleren of metingen te checken. Zo worden bijvoorbeeld uit de meetwaarden van *MEETDUUR*, *IN*, *UIT* en *R* (of *Z*) de quotiënten R/UIT, $(s \times MEETDUUR - R)/IN$ en $UIT/MEETDUUR$ gevormd. Als die quotiënten worden ingevuld in de responsetijdrelatie voor de overall gemiddelden, waarin nog niet $IN = UIT$ is gesteld, resulteert er exact een identiteit. Invullen in de responsetijdrelatie waarin wel $IN = UIT$ is gesteld levert vrijwel een identiteit als *IN* en *UIT* niet al te klein zijn ten opzichte van *s*. Een meting levert zo geen bevestiging of verwerping van responsetijdrelatie-'wetten', ze zijn zonder meer juist.

De *'uitgebreide'* vormen van de responsetijdrelaties zijn relaties tussen verwachtingswaarden van stochasten die op het systeem betrekking hebben. De voor operationele schatters gevonden relaties zouden ook gelden voor de 'echte' gemiddelden. Deze 'echte' responsetijdrelaties luiden

som van gemiddelde responsetijd en gemiddelde denktijd
=
aantal gebruikers / (gemiddelde doorstroom)

waarbij onder gemiddelde moet worden verstaan de betreffende verwachtingswaarde†.

De uitgebreide vorm van de responsetijdrelaties is ook heel ruim toepasbaar. Bij de interpretatie van de relatie van Little uit het vorige hoofdstuk merkten we al op dat de 'echte', de 'uitgebreide' vorm daarvan, *'onbeperkt'* toepasbaar is. We zullen de responsetijdrelaties aanstonds terugbrengen tot een herhaald toepassen van de relatie van Little.

In benaderende doorrekenbare modellen, die in termen van verdelingen van stochastische variabelen gedefinieerd zijn, wordt steeds de uitgebreide vorm van de

---

† De inverse van de doorstroom is de gemiddelde tussenpoos tussen afgewerkte opdrachten.

responsetijdrelatie aangetroffen. Onze bevindingen hier maken al duidelijk dat de responsetijdrelaties ook buiten deze modellen geldigheid hebben.

5.4.1. voorbeeld: triviale opdrachten

Voor simpele 'triviale' opdrachten garandeert een rekencentrum een responsetijd, die in 90% van de gevallen niet hoger is dan 0.2 sekonden. De gemiddelde responsetijd voor deze opdrachten blijkt 0.1 sekonden te zijn. De doorstroom aan triviale opdrachten is als er 50 terminalisten mee bezig zijn gemiddeld 25 opdrachten per sekonde. Wat is de gemiddelde denktijd bij triviale opdrachten?

De responsetijdrelatie zegt dat tussen de gemiddelden, hoe ook opgevat, de volgende betrekking geldt:

$$0.1 = 50/25 - \text{gemiddelde denktijd} \rightarrow \text{gemiddelde denktijd} = 1.9 \text{ sek.}$$

5.5. TOEPASBAARHEID RESPONSETIJDRELATIES

De beperkte vorm van de responsetijdrelaties gaan we als leidraad gebruiken bij het bespreken van globale verschijnselen, die zich voordoen als het bij een computercentrum drukker wordt. We bekijken een heel eenvoudige vraagstelling om te zien of de ingevoerde begrippen inderdaad relevant zijn.

5.5.1. voorbeeld: verbetering service

Bij vele rekencentra wordt waargenomen dat wanneer de turnaround-tijd omlaag wordt gebracht, de gebruikers meer jobs per tijdseenheid gaan inleveren. Het is niet zo dat het aantal jobs (die gemiddeld eerder klaar zijn) per bepaalde periode constant blijft. Anderzijds wordt waargenomen dat wanneer de gebruikers meer jobs per tijdseenheid gaan aanbieden, de turnaround-tijd gaat stijgen. Soms lijkt het alsof er iets paradoxaals in deze waarnemingen zit. Spelen er psychologische effekten een rol?

De doorstroom van een systeem kan nooit willekeurig groot worden, er is altijd een bovengrens aan het aantal jobs, van het soort dat door de gebruikers zoal wordt aangeboden, dat door het systeem per tijdseenheid kan worden verwerkt.

Veronderstel dat de gebruikers een bepaalde tijdsduur L mogen benutten om UIT opdrachten aan te bieden en verwerkt terug te krijgen (we hanteren onze standaardterminologie met opdrachten en responsetijd). De machine en de gebruikers moeten al die tijd optimaal werken om deze opdrachten er door te krijgen. De machine kan hoogstens gemiddeld $\max_{oud}$ van dit soort programma's per tijdseenheid verwerken, dit is aanvankelijk de maximale doorstroom. Voor deze tijdsduur L geldt als de gebruikers hun job klaren, dat exact $IN = UIT$. De responsetijdrelatie voor deze 'meetduur' luidt, we gebruiken om onderscheid te

maken de onderindices $_{oud}$

$$'R_{oud} = s / \max_{oud} - 'Z_{oud} \qquad \Rightarrow \text{(relatie } oud\text{)}$$

Telkens als de gebruikers weer een tijd L de tijd krijgen, realiseren ze een 'pad door de wereld' van mogelijkheden voor het verloop van de operatie. De per keer (per pad) waargenomen gemiddelde responsetijd, doorstroom en denktijd, dat zijn de gemeten waarden van de operationele schatters, voldoen steeds aan deze relatie. We zullen overigens voortaan de lange aanduiding 'operationele schatter voor' in de benaming van de grootheden uit de beperkte vorm van een responsetijdrelatie maar weglaten, als er geen verwarring mogelijk is.

Op zeker moment lukt het de systeembeheerders om door een betere tuning de maximaal mogelijke doorstroom op te voeren tot de waarde $\max_{nieuw}$. Nadat deze wijziging is doorgevoerd breekt er opnieuw een periode met duur L aan, waarin de UIT opdrachten worden verwerkt. De gebruikers zijn —als ze vanaf het begin even hard en geconcentreerd werken als vroeger— eerder klaar. Daarna luieren ze de rest (LUI) van de tijdsduur L. In de nieuwe situatie is de responsetijdrelatie voor de duur L

$$'R_{nieuw} = s / \max_{oud} - 'Z_{nieuw} \qquad \Rightarrow \text{(relatie } nieuw\text{)}$$

De gemiddelde doorstroom is niet anders dan in de vroegere situatie, hij was $UIT / L = \max_{oud}$ en dat is hij nog steeds. Over de tijdsduur $AKTIEF$ waarin de gebruikers stug werken geldt echter

$$'R_{aktief} = s / \max_{nieuw} - 'Z_{aktief} \qquad \Rightarrow \text{(relatie } aktief\text{)}$$

De machine werkt hier op zijn top, de doorstroom is $\max_{nieuw}$, deze is groter dan $\max_{oud}$ uit de oude situatie. Omdat de machine de UIT opdrachten in de tijdsduur $AKTIEF$ verwerkt, is $LUI = L - UIT / \max_{nieuw}$, want $\max_{nieuw} = UIT / AKTIEF$. De gemiddelde denktijd zal in alle perioden van geconcentreerd werken gelijk zijn (de denktijd wordt voornamelijk bepaald door vaste administratieve en reaktieve faktoren), zodat de waarden voor $'Z_{oud}$ en $'Z_{aktief}$ meestal niet veel uiteenlopen. De waarden voor $'R_{aktief}$ zijn daardoor zo'n $s / (UIT / LUI)$ lager dan die voor $'R_{oud}$.

We vergelijken de drie responsetijdrelaties 'oud', 'nieuw' en 'aktief'.

In de periode van hard werken is de gemiddelde responsetijd lager dan in de oude situatie. Alle responsen zijn al afgelopen als het luieren begint. Er verandert daarna niets meer aan de tijd waarin op response wordt gewacht, dus $R_{nieuw} = R_{aktief}$ en $'R_{nieuw} = 'R_{aktief}$. In de nieuwe situatie is dus ook over de hele tijdsduur gerekend de gemiddelde responsetijd lager dan in de oude; het linkerlid van relatie

'nieuw' is $'R_{aktief}$. De relatie 'nieuw' is dus ook:

$$'R_{aktief} = s / \max_{oud} - 'Z_{nieuw} \qquad \Rightarrow \text{(relatie } nieuw\text{)}$$

De relaties 'nieuw' en 'aktief' verschillen dus in gemiddelde doorstroom en gemiddelde denktijd.

De gemiddelde denktijd over de hele tijdsduur genomen (in relatie 'nieuw') is iets heel anders dan de gemiddelde denktijd in de periode van hard werken (in relatie 'aktief'). De totale denktijd Z in de nieuwe situatie bestaat uit de som van alle denktijden in de periode van hard werken, plus voor iedere gebruiker een aaneengesloten denkduur van LUI, zodat in relatie 'nieuw' $'Z_{nieuw} = 'Z_{aktief} + s \times LUI / UIT$. In relatie 'nieuw' wordt ook het luieren als denken meegenomen! Het verschil in doorstroom tussen 'nieuw' en 'aktief' ($\max_{oud}$ versus $\max_{nieuw}$) compenseert precies dit oneigenlijke deel van de gemiddelde denktijd, immers $s / \max_{oud} - s / \max_{nieuw} = s \times LUI / UIT$.

Keurig formeel vasthouden aan de duur L als referentie en het zien van luieren als denken is kennelijk volledig gerechtvaardigd; het is echt altijd mogelijk als meetduur een duur L vanaf het begin van het werk te nemen. Maar het zou hier natuurlijk verstandig zijn zich direkt te beperken tot de tijd $AKTIEF$ waarin gewerkt wordt en de relatie 'aktief' te propageren, niet de relatie 'nieuw'.

De aangegeven onderlinge samenhang tussen de responsetijdrelaties verandert niet als het luieren zich niet uitsluitend aan het eind van de meetduur afspeelt, maar ook zo tussendoor wel eens minder hard gewerkt wordt en de opdrachten pas veel later dan na de tijdsduur $AKTIEF$ verwerkt zijn. Luieren en denken zijn dan niet meer te scheiden en de gemiddelde denktijd kan worden verlaagd als men zich inspant (geconcentreerd werken houdt ook in zich telkens instellen op de verwachte duur van de responsetijd van een ingebrachte opdracht, zodat steeds tijdig met het nodige denkwerk wordt begonnen; luieren is wel een heel gebrekkige naamgeving; zie ook §6.3).

De gebruikers zullen heel gauw met werk worden opgezadeld dat ze kunnen doen in de tijd waarin ze luieren. Als de toevloed aan werk hoog genoeg is, dwingen ze de machine al snel weer tot de maximale prestatie over de hele meetduur L en geldt ook over de hele duur (de waarden voor $'Z_{nieuw}$ zullen weinig verschillen van die voor $'Z_{oud}$)

$$'R_{nieuw} = s / \max_{nieuw} - 'Z_{nieuw}$$

Het aantal in de duur L verwerkte opdrachten neemt daarbij toe van $UIT = \max_{nieuw} \times AKTIEF$ tot gemiddeld $\max_{nieuw} \times L$.

De betere tuning heeft dus bewerkstelligd dat de gemiddelde responsetijd (of turnaround-tijd) is gedaald ($'R_{nieuw} < 'R_{oud}$). Bovendien worden er gemiddeld meer opdrachten (jobs) aangeboden ($\max_{nieuw} \times L > UIT$). Deze beide constateringen stemmen overeen met de vermelde waarnemingen uit de praktijk.

Alle verklaringen zijn nog heel algemeen. Ze gelden zowel voor batchgebruikers als voor terminalisten. Specificaties zoals de bedieningsregeling (bijvoorbeeld time-sharing) zijn niet nodig. Natuurlijk geven zulke 'oppervlakkige' globale relaties slechts iets aan over gemiddelden. In de praktijk kan het funest zijn af te zien van de verdelingen rond gemiddelden, wanneer eisen van gebruikers gehonoreerd moeten worden. De resultaten van verdere analyse na detaillering kunnen echter niet ingaan tegen wat de responsetijdrelaties leren.

5.5.2. voorbeeld: beperkte capaciteit

Laten we ons eens indenken, dat de gebruikers proberen om nog sneller hun klus af te ronden: ze vragen of huren vrienden en kennissen om vanaf het begin van een periode L mee te werken de UIT opdrachten verwerkt te krijgen. Het effekt dat ze hiermee verkrijgen is misschien dramatisch: het duurt met de extra helpers gemiddeld even lang als zonder hen, pas na gemiddeld evenveel tijd is de klus geklaard. De maximaal mogelijke doorstroom werd al gehaald in de tijdsduur $AKTIEF$ vanaf het begin. Het aantal in die tijdsduur verwerkte opdrachten is gemiddeld $UIT = \max_{nieuw} \times AKTIEF$, zo'n aantal zal niet hoger opgevoerd kunnen worden.

Wat merken de gebruikers van het meehelpen van de anderen? In de responsetijdrelatie over de tijd $AKTIEF$ van hard werken verandert na de komst van s_{hulp} helpers de term $s / \max_{nieuw}$ in $(s + s_{hulp}) / \max_{nieuw}$. De gemiddelde denktijd verandert niet, want alle gebruikers, ook de extra gebruikers, werken geconcentreerd. De responsetijdrelatie zegt dan dat de gemiddelde responsetijd zal toenemen met $s_{hulp} / \max_{nieuw}$. Alle gebruikers, zowel helpers als geholpenen, hebben te maken met deze daling in response.

Aan de responsetijdrelatie uit §5.3 voor iedere gebruiker apart

$$'R_A = 1 / 'doorstroom_A - 'Z_A$$

lezen we af dat bij verhoging van zijn gemiddelde responsetijd ($'R_A$) de eigen doorstroom ($'doorstroom_A$) daalt. De totale doorstroom in de tijd van aktief werken blijft gelijk, maar individuele bijdragen daarin dalen noodgedwongen.

Dit eenvoudige voorbeeld laat een fenomeen zien, dat altijd optreedt als een grens voor de verwerkingssnelheid wordt bereikt: een collectieve verbetering valt niet meer te behalen. De machine is ***knelpunt*** geworden.

5.6. WISSELEND AANTAL GEBRUIKERS

We veronderstelden in onze responsetijdrelaties tot nu toe dat er eenvoudig constant s gebruikers tijdens de meetduur bezig zijn, die afwisselend wachten en denken. Het is niet moeilijk de relaties uit te breiden naar meetduren, waarin het aantal gebruikers varieert.

We noemen onze gebruikers aangesloten op het systeem. Een terminalist die een poos lang geen aktiviteiten onderneemt, doordat hij bezoek heeft gekregen of vergeten is uit te loggen of suft en slaapt, is wel ingelogd maar niet aangesloten; onder de aangesloten gebruikers moeten de aktief betrokkenen worden verstaan. Alle tijdens een meetduur ooit aangesloten gebruikers kunnen natuurlijk geacht worden de hele meetduur aanwezig te zijn, er is dan steeds een vast aantal gebruikers maar tijdens het niet-aangesloten zijn ontstaan oneigenlijke bijdragen tot de denktijd, die gecompenseerd worden door bijdragen in de term s/doorstroom; we zagen dit zojuist bij gebruikers die ver binnen de meetduur hun karwei geklaard hebben. Zo'n beschrijving is onnatuurlijk; alleen in een beschrijving met een expliciet wisselend aantal aangesloten gebruikers kan de denktijd zijn eigenlijke betekenis van reaktietijd behouden. Variaties in het aantal gebruikers ontstaan bijvoorbeeld wanneer een bepaalde terminal een tijdlang onbezet is, of in een formulering die ook toepasbaar is op batchgebruikers: wanneer een bepaalde lijn van elkaar opvolgende individuele gebruikers een tijdlang leeg is.

Een gebruiker kan verder soms met een aantal opdrachten tegelijk bezig zijn, zo kan een terminalist een commando op de achtergrond laten uitvoeren en ondertussen op de normale manier verder werken. Als de gebruiker zijn scherm kan verdelen in windows, zal hij tijdens het verwerken van een opdracht misschien 'overclicken' naar een ander venster en daar doorgaan. Al zulke extra ingebrachte opdrachten die de mooie afwisseling van denk- en responsetijd verbreken, kunnen we toeschrijven aan fiktieve extra gebruikers. Het effektief aantal aktieve gebruikers is de som van de ingelogde gebruikers plus voor elke opdracht 'op de achtergrond' een fiktieve gebruiker. Dit effektieve aantal gebruikers varieert dus in de tijd. De totale tijd door aangesloten gebruikers doorgebracht, wordt aangegeven door de collectieve grootheid S. Eerder, in het hoofdstuk DE RELATIE VAN LITTLE (§4.2), noemden we de tijd door de gebruikers ingelogd doorgebracht ook S. We gebruiken hier hetzelfde symbool voor een rechtstreekse veralgemening van die grootheid.

De operationele schatter voor het gemiddeld aantal gebruikers, dat in de meetduur $MEETDUUR$ aangesloten is, luidt dan ook

$$'s = S/MEETDUUR$$

De totale aangesloten tijd wordt òf in de toestand 'in response' òf in de toestand 'denkend' doorgebracht, zodat voor de collectieve grootheden R en Z volgt

$$S = R + Z$$

Substitutie hierin van $'R = R/UIT$, $'Z = Z/IN$, $'doorstroom = UIT/MEETDUUR$ en $'s = S/MEETDUUR$ levert na herschrijven

$$'R = 's / 'doorstroom - (IN / UIT)\, 'Z$$

Dit is de nieuwe uitdrukking voor de overall responsetijdrelatie. De veralgemening uit zich in niet meer dan het vervangen van het aantal gebruikers *s* door het gemiddelde daarvan: *'s*.

5.6.1. soorten gebruikers

Gebruikers kunnen op vele manieren groepsgewijs worden onderscheiden of samengenomen, bijvoorbeeld op basis van een bepaald gemeenschappelijk kenmerk als prioriteit of CPU-verbruik, of in editorgebruikers, compilergebruikers, gebruikers van een bepaalde terminal, enzovoort. De beschrijving van deze opsplitsing is eenvoudig wanneer er operationele schatters worden gehanteerd. Alle grootheden en operationele schatters die betrekking hebben op de prestaties van gebruikers van *groep g* voorzien we van de index *g* als specificatie. Zo is S_g de collectieve grootheid die de totale tijd weergeeft door gebruikers van groep *g* aangesloten doorgebracht. Deze bestaat weer uit een totale responsetijd R_g en een totale denktijd Z_g, met $S_g = R_g + Z_g$. Invullen van de operationele schatters voor gebruikers van groep *g* levert

$$'R_g = 's_g / 'doorstroom_g - (IN_g / UIT_g)\, 'Z_g$$

Dit is de responsetijdrelatie voor gebruikers van groep *g*.

Ook alle gemiddelden zijn op te splitsen naar groepsbijdragen, de gewichten bij samenstelling tot overall grootheden volgen rechtstreeks. Zo is in de eerste plaats natuurlijk (de som is over de index *g* die alle groepen doorloopt)

$$'doorstroom = UIT / MEETDUUR = \sum_g UIT_g / MEETDUUR$$

of

$$'doorstroom = \sum_g 'doorstroom_g$$

Bij de overall (gemiddelde) doorstroom wordt niet gemiddeld over de groepen, maar iedere groep draagt zijn eigen doorstroom bij in de 'overall' doorstroom. Er vindt echt geen weging plaats. De specificatie gemiddeld in 'operationele schatter voor gemiddelde doorstroom' wekt misschien toch de verwachting van een middeling met gewichten, hoewel deze toevoeging slaat op de middeling over de variërende omstandigheden. We laten ook daarom —als in de literatuur— standaard het voorvoegsel gemiddeld bij doorstroom weg (§5.2).

Voor de overall gemiddelde responsetijd loopt de middeling zo:

$$'R = R/UIT = \sum_g R_g/UIT$$

$$= \sum_g (R_g/UIT_g)(UIT_g/MEETDUUR)/(UIT/MEETDUUR)$$

of

$$'R = \sum_g ('doorstroom_g/'doorstroom)\,'R_g$$

De gemiddelde responsetijd $'R_g$ van de gebruikers van een bepaalde groep draagt bij in de overall gemiddelde responsetijd $'R$. Het gewicht van zijn bijdrage is, zo staat hier, het relatieve aandeel van gebruikers van deze groep in de doorstroom. Bij middeling van groepsgemiddelden tot een 'overall' gemiddelde treedt heel vaak deze gewichtsfactor op. De middeling van de responsetijd is duidelijk niet volgens het aantal terminalisten in de groep, zoals wellicht op het eerste gezicht zou worden gedacht. Er wordt —in de terminologie uit het komende hoofdstuk BEURTEN EN KANSEN — onder de opdrachten gemiddeld naar kans op optreden.

Met naar groep opgesplitste relaties (zoals de voorgaande) kunnen de effekten van groepen tijdelijk aangesloten extra gebruikers worden besproken. De beslissing wanneer het laatste denken ophoudt en het niet-aangesloten zijn begint, is natuurlijk arbitrair en moet redelijk genomen worden. In het extreme geval is een gebruiker alleen aangesloten als hij in de responsefase is. Voor een groep waarvoor dit het geval is (dit zouden bijvoorbeeld achtergrondopdrachten kunnen zijn) geldt

$$'R_g = 's_g/('doorstroom_g)$$

Dan is echter $'s_g$ ook het gemiddeld aantal gebruikers van deze groep in de toestand 'in response', we krijgen voor deze groep gewoon een geval van de relatie van Little terug. Men noemt zo'n groep wel een 'open' groep. Werk van gewone interaktieve gebruikers is werk van een 'gesloten' groep (vergelijk §14.8). Voor een gesloten groep geldt een responsetijdrelatie, voor een open groep niet (§6.4).

5.7. VOORBEELD: OPERATIONELE SCHATTERS TERMINALBEZETTING

We rekenen een concrete waarneming door. Er zijn 6 terminals op een computersysteem aangesloten. Denk- en responsetijden werden gemeten gedurende een meetduur van 3000 sekonden. We geven het begin van deze meetduur aan als (tijdstip) 0 en het eind als (tijdstip) 3000. De terminals 1 en 2 zijn ingelogd geweest van 0 tot 1000 en van 1500 tot 3000 en de terminals 3, 4 en 5 de gehele meetduur. Terminal 6 was slechts van tijdstip 1000 tot tijdstip 2500 ingelogd. De

gebruikers aan de terminals 1 en 2 blijken hun ingelogde tijd steeds voor driekwart met 'denken' door te brengen. Aan de terminals 3, 4 en 5 is de helft van de ingelogde tijd in de denkfase doorgebracht. De terminalist aan terminal 6 was de helft van de tijd denkend. In het tijdinterval 0-1000 zijn voor elk van de terminals 1-5 50 ingetoetste opdrachten verwerkt en in het tijdinterval 1000-3000 voor elk van de terminals 1-5 100 ingetoetste opdrachten en 50 voor terminal 6. De gemeten waarden staan in tabel 5.2.

$$\begin{aligned} UIT_1 \text{ t/m } UIT_5 &= 50+100 = 150 \text{ opdr.} \\ UIT_6 &= 50 \text{ opdr.} \\ Z_1 = Z_2 &= (1000+1500)\times 3/4 = 1875 \text{ sek.} \\ Z_3 \text{ t/m } Z_5 &= 3000/2 = 1500 \text{ sek.} \\ Z_6 &= 1500/2 = 750 \text{ sek.} \\ R_1 = R_2 &= 2500/4 = 625 \text{ sek.} \\ R_3 \text{ t/m } R_6 &= Z_3 \text{ t/m } Z_6 \end{aligned}$$

Tabel 5.2. Meetwaarden terminalbezetting.

Over de hele meetduur gerekend zijn de waarden voor de operationele schatters $'Z$, $'R$, $'s$ en *'doorstroom* voor alle terminals samen, met $IN = UIT = 5\times 150+50 =$ 800 opdrachten:

$$'Z = (2\times 1875+3\times 1500+750)/800 = 45/4 \text{ sek.}$$

$$'R = (2\times 625+3\times 1500+750)/800 = 65/8 \text{ sek.}$$

$$'s = (2\times 2500+3\times 3000+1500)/3000 = 31/6 \text{ terminalisten}$$

$$'doorstroom = 800/3000 = 4/15 \text{ opdr./sek.}$$

Inderdaad is

$$65/8 = (31/6)/(4/15) - 45/4$$

Er zijn, omdat er niet continu op elke terminal is ingelogd, gemiddeld slechts 5 1/6 terminalisten aktief. De gemeten gemiddelde responsetijd aan de terminals 1 en 2 samengenomen, aan de terminals 3 tot en met 5 samengenomen en die aan ter-

minal 6 zijn (in sekonden)

$$'R_{12} = (2\times625)/300 = 25/6$$

$$'R_{345} = (3\times1500)/450 = 10$$

$$'R_6 = 750/50 = 15$$

De terminalisten aan terminal 1 hadden gemiddeld een doorstroom aan opdrachten van (per sekonde)

$$'doorstroom_1 = 150/3000 = 1/20$$

Hetzelfde bij terminal 2. De andere gemeten waarden voor de doorstromen zijn (aan de terminals 3 tot en met 5 was de (gemiddelde) doorstroom gelijk)

$$'doorstroom_3 = 150/3000 = 1/20$$

$$'doorstroom_6 = 50/3000 = 1/60$$

De overall gemiddelde responsetijd van 65/8 sekonden is een gewogen gemiddelde van de gemiddelde responsetijden 25/6, 10 en 15 sekonden voor de diverse groepen terminals. De weging is naar de bijdragen in de overall doorstroom, volgens

$$65/8 = (25/6\times2/20+10\times3/20+15/60)/(5/20+1/60)$$

De waarde van de schatter voor de gemiddelde denktijd aan de terminals 1 en 2 samen is

$$'Z_{12} = 2\times1875/300 = 25/2$$

Het gemiddeld aantal ingelogde terminalisten is voor deze terminals over de totale meetduur gezien

$$'s_{12} = 2\times2500/3000 = 5/3$$

De responsetijdrelatie voor deze terminals is dus

$$25/6 = (5/3)/(2/20) - 25/2$$

Maar natuurlijk is het, als men zich tot slechts deze terminals wil beperken, verstandiger te zeggen dat er $150 \times 2 = 300$ opdrachten verwerkt worden in 2500 sekonden, dus dat de gemiddelde doorstroom bij bezette terminal $300/2500 = 0.12$ opdrachten per sekonde is. De responsetijdrelatie wordt dan

$$25/6 = 2/(3/25) - 25/2$$

Voor het gemiddeld aantal terminalisten moet nu natuurlijk precies 2 worden genomen. Ook in de afspraken voor gemiddeld aantal terminalisten en (gemiddelde) doorstroom is het kennelijk zaak om afhankelijk van de omstandigheden een relevante keus te maken.

Aan de terminals 1 en 2 wordt waarschijnlijk heel ander werk gedaan dan bij de terminals 3 tot en met 5. De responsetijden verschillen sterk en ook de denktijden lopen uiteen. De 'overall' waarden zijn in zo'n geval niet voldoende om de situatie ook maar globaal te beschrijven. Maar men ziet hier heel duidelijk hoe er fatsoenlijk gemiddeld moet worden, zowel in de tijd als over de terminals. Aan terminal 6 worden in een meetduur veel minder waarnemingen gedaan dan aan een andere terminal. Dat wordt keurig in rekening gebracht.

5.8. TOEPASSINGEN RESPONSETIJDRELATIES

De operationele schatters geven entree tot de 'echte' (de 'uitgebreide') vorm van de responsetijdrelaties.

We verlaten nu deze toegangspoort en bepalen ons in het vervolg tot de echte relaties. Alleen wanneer we willen weten hoe een bepaalde opsplitsing precies verloopt, doen we een stapje terug naar de operationele schatters.

5.8.1. voorbeeld: editeren/compileren

De gemiddelde responsetijd voor editcommando's is 0.5 sekonden, de gemiddelde denktijd bij het editen is 2 sekonden. De gemiddelde responsetijd voor compilaties is 2 sekonden, de gemiddelde denktijd daarbij is 10 sekonden. Ingebrachte opdrachten bestaan voor 92% uit editcommando's, slechts 8% zijn compileercommando's. De (overall) doorstroom is hier opgebouwd uit twee bestanddelen (twee soorten gebruikers):

$$doorstroom = doorstroom_{edit} + doorstroom_{comp}$$

De verhouding tussen de in een meetduur te meten gemiddelde aantallen editcommando's en compileercommando's zal zijn 92/8. Aan de vorm van de operationele schatters voor de (gemiddelde) doorstroom valt dan meteen af te lezen, dat de verhouding tussen de overall doorstroom en de doorstroom aan editcommando's is

$$\frac{doorstroom_{edit}}{doorstroom} = 0.92$$

Dit is de passende vertaling van het gegeven dat de opdrachten voor 92% uit editcommando's bestaan; zo'n informatie zegt iets over de samenstelling van de werklast en daarmee over de relatieve doorstromen. De gegevens slaan op de 'echte' gemiddelden, deze geven we aan met het voorvoegsel *gem.*; voor de 'echte' overall doorstroom schrijven we kortweg *d*, in plaats van doorstroom.

Uit de responsetijdrelaties voor edit(eer)- en compileeropdrachten (we hanteren de uitgebreide vorm en gebruiken dan ook geen aanhalingstekens)

$$0.5 = \frac{gem.\ s_{edit}}{0.92 \times d} - 2$$

$$2 = \frac{gem.\ s_{comp}}{0.08 \times d} - 10$$

volgt na eliminatie van *d*

$$\frac{gem.\ s_{edit}}{gem.\ s_{comp}} = 2.3 / 0.96$$

Het gemiddeld aantal terminalisten dat zich bezig houdt met editen is maar een fractie 2.3/3.26 (= 0.71) van het gemiddeld aantal aktieve terminalisten. Dit is aanzienlijk minder dan het aandeel van 92% van de editcommando's in de doorstroom.

De gemiddelde overall responsetijd is volgens §5.6.1

$$gem.\ responsetijd = 0.92 \times 0.5 + 0.08 \times 2 = 0.62$$

De gemiddelde overall denktijd is idem

$$gem.\ denktijd = 0.92 \times 2 + 0.08 \times 10 = 2.64$$

Bij een overall doorstroom van 10 opdrachten per sekonde is

$$gem.\ s_{edit} = 2.5 \times 9.2 = 23$$

$$gem.\ s_{comp} = 9.6$$

zodat

$$gem.\ s = 32.6$$

Inderdaad geldt, hoe verschillend editen en compileren ook zijn, voor de overall gemiddelde waarden de responsetijdrelatie:

$$0.62 = 32.6/10 - 2.64$$

5.8.2. voorbeeld: tolwegen

Het rekenwerk bij het splitsen naar soorten is bekend en vertrouwd. Neem bijvoorbeeld een autosnelweg, die per uur 1200 auto's verwerkt. Om op de snelweg te komen moet een auto een 'loket' passeren en tol betalen. Een derde deel van de auto's rijdt langs loket 3, een zesde deel langs loket 6. Een derde van de auto's die aan deze loketten tol betalen is rood, de anderen zijn grijs. Rode auto's wegen gemiddeld 0.6 ton, grijze auto's 0.3 ton.

De (gemiddelde) doorstroom aan auto's door loket 3 is 400 auto's per uur. De doorstroom aan auto's, die door loket 3 of loket 6 zijn gegaan, is 400+200 = 600 auto's per uur. Deze auto's wegen gemiddeld per stuk $1/3 \times 0.6 + 2/3 \times 0.3 = 0.4$ ton, dus het overall gemiddelde gewicht is 0.4 ton. Als de grijze auto's gemiddeld tweemaal zo hard rijden als de rode, is volgens Little het gemiddeld aantal rode auto's op de weg, die via loket 3 of loket 6 op de weg kwamen, even groot als het gemiddeld aantal grijze auto's, die er via die loketten kwamen. Enzovoort.

5.8.3. voorbeeld: commando's op de achtergrond

De doorstroom aan commando's, uitgevoerd door de commando-interpretator (sh) —de systeemcommando's—, bestaat voor 5% uit commando's op de achtergrond. Een terminalist kan na het starten van een commando op de achtergrond direkt verder gaan met het inbrengen van commando's, er hoeft niet zoals gewoonlijk te worden gewacht op de gereedmelding (prompt). De gemiddelde 'real time' van een systeemcommando op de achtergrond, dat is de tijdsduur tussen de start als 'achtergrondcommando' en het gereedkomen, bedraagt 57 sekonden. De terminalisten brengen gemiddeld 1/19 achtergrondcommando's per sekonde in. Er zijn in het systeem meestal een aantal systeemcommando's aanwezig, die op de

achtergrond draaien. Het zijn er gemiddeld 3 volgens Little:

$$(\textit{doorstroom aan achtergrondcommando's} = 1/19) \times 57 = 3$$

Naast de achtergrondcommando's worden door de terminalisten ook de gewone 'interaktieve' systeemcommando's ingebracht, die meestal het merendeel van de opdrachten uitmaken. De gemiddelde denktijd bij interaktieve systeemcommando's is 10 sekonden, er zijn gemiddeld 15 terminalisten die dergelijke commando's inbrengen. Ze genereren een doorstroom van gemiddeld 1 systeemcommando per sekonde.

De gemiddelde responsetijd voor interaktieve systeemcommando's is

$$\textit{gem. responsetijd} = 15/1 - 10 = 5 \text{ sek.}$$

Het gemiddeld aantal van deze commando's in de responsefase is volgens Little

$$\textit{gem. aantal in response} = 1 \times 5 = 5$$

De totale doorstroom aan systeemcommando's is $1/19+1 = 20/19$ per sekonde. Hiervan is in dit geval $1/20$ deel achtergrondcommando. In het systeem zitten gemiddeld 3 achtergrondcommando's en gemiddeld 5 interaktieve commando's. Hoewel de achtergrondcommando's maar een gering deel van de doorstroom uitmaken (5%), dragen ze sterk bij in het gemiddeld aantal systeemcommando's in het systeem ($3/8 \times 100 = 37.5\%$), omdat ze zo lang in het systeem vertoeven. Ze zitten er gemiddeld 57 sekonden en een interaktief commando maar 5 sekonden.

5.9. RESPONSETIJDRELATIES EN RELATIE VAN LITTLE

Bij de relatie van Little merkten we op dat deze kan worden toegepast wanneer deelsystemen parallel in de tijd telkens eenzelfde toestand bezetten. De terminalisten funktioneren parallel in de tijd en elk deelsysteem 'terminalist' komt telkens in de toestand 'denkend'. Voor de operationele schatter voor het gemiddeld aantal terminalisten (deelsystemen) in de denkfase gebruiken we de notatie $'z$, idem voor het gemiddeld aantal in de responsefase $'r$.
De relatie van Little leert met†

$$\textit{gem. aantal} = \textit{gem. aantal gebruikers denkend} \leftarrow {'z}$$

† 'Ongeveer gelijk aan' wordt aangegeven door $\approx$, correspondeert met door $\leftarrow$.

$$\textit{gem. duur} = \textit{gem. denktijd} \leftarrow {}'Z$$

$$\textit{gem. stroom} \leftarrow IN\,/\,MEETDUUR \approx {}'\textit{doorstroom}$$

dat (voor bijvoorbeeld de beperkte vorm)

$$'z = {}'\textit{doorstroom} \times {}'Z$$

Het gemiddelde aantal denkende terminalisten is gelijk aan de doorstroom maal de gemiddelde denktijd.

We kunnen met evenveel recht beweren dat het deelsysteem 'terminalist' telkens in de toestand 'in response' komt. Ook daarbij moet de relatie van Little gelden, met nu

$$\textit{gem. aantal} = \textit{gem. aantal in response} \leftarrow {}'r$$

$$\textit{gem. duur} = \textit{gem. responsetijd} \leftarrow {}'R$$

$$\textit{gem. stroom} \leftarrow UIT\,/\,MEETDUUR = {}'\textit{doorstroom}$$

of idem

$$'r = {}'\textit{doorstroom} \times {}'R$$

Het gemiddelde aantal terminalisten in de responsefase is gelijk aan de doorstroom maal de gemiddelde responsetijd. Optellen van de beide relaties geeft

$$'r + {}'z = {}'\textit{doorstroom} \times ({}'R + {}'Z)$$

En omdat een terminalist òf denkt òf in response is, geldt

$$'s = {}'r + {}'z$$

zodat

$$'s = {}'\textit{doorstroom} \times ({}'R + {}'Z)$$

Het gemiddeld aantal terminalisten is gelijk aan de doorstroom maal de som van gemiddelde responsetijd en gemiddelde denktijd. Dat is gewoon een herschreven vorm van de meest algemene responsetijdrelatie (in de 'beperkte' vorm genoteerd).

We zien zo dat de responsetijdrelaties geschreven kunnen worden als het tweemaal toepassen van de relatie van Little, eenmaal op de toestand 'denkend' en eenmaal op de toestand 'in response' ofwel eenmaal vanuit de kant 'gebruikers' en eenmaal vanuit de kant 'computersysteem'.

We kunnen ook zeggen dat de deelsystemen 'terminalist' telkens een cycle volbrengen van 'in response, denkend'. De responsetijdrelatie in de bovenstaande vorm 'gemiddeld aantal terminalisten is gelijk aan de doorstroom maal de som van gemiddelde responsetijd en gemiddelde denktijd' kan daarom op zich simpelweg gezien worden als een direkte toepassing van de relatie van Little, met $'s$ of *gem. s* als het gemiddeld aantal deelsystemen dat steeds deze cycle uitvoert, de doorstroom als de stroom aan cycles en met als duur de cycleduur, dat is de som van responsetijd en denktijd. Inderdaad zijn de responsetijdrelaties niets anders dan een toepassing van de relatie van Little.

Het gemiddeld aantal terminalisten in de responsefase is natuurlijk ook het gemiddeld aantal in het systeem ingebrachte, maar nog niet geheel verwerkte opdrachten. Zulke opdrachten staan ergens in het systeem in een wachtrij of worden door een van de devices van het systeem verwerkt. De grootheden 'gemiddeld aantal in response' (à la $'r$) en 'gemiddeld aantal denkend' (à la $'z$) zijn heel nuttige globale grootheden, men komt ze vaak tegen.

5.9.1. vervolg voorbeeld: denk/responsetijd (5.3.1)

De gemeten waarde van de operationele schatter voor het gemiddeld aantal opdrachten in de responsefase is

$$'r = '\textit{doorstroom} \times 'R = 402/1000 \times 7600/402 = 7.6$$

Het gemiddeld aantal in de denkfase is idem

$$'z = 2400/1000 = 2.4$$

Van de 10 terminalisten denken er gemiddeld 2.4.

5.9.2. vervolg voorbeeld: triviale opdrachten (5.4.1)

Naast de triviale opdrachten worden ook 'zware' opdrachten ingebracht. Deze hebben een gemiddelde responsetijd van 1 sekonde. Door 10 terminalisten worden gemiddeld per sekonde 2.5 zware opdrachten ingebracht. De responsetijdrelatie voor deze zware opdrachten zegt dat de gemiddelde denktijd is 3 sekonden, want

$$1 = 10/2.5 - 3$$

Het gemiddeld aantal zware opdrachten in de responsefase is $2.5 \times 1 = 2.5$

opdrachten. De doorstroom aan triviale opdrachten was 25 per sekonde bij een gemiddelde responsetijd van 0.1 sekonden. Er zijn dus gemiddeld $25 \times 0.1 = 2.5$ triviale opdrachten in de responsefase. Hoewel de doorstroom aan triviale opdrachten een factor 10 groter is dan die aan zware opdrachten —er worden veel meer triviale opdrachten verwerkt— zijn er intern in het systeem gemiddeld evenveel triviale als zware opdrachten aanwezig. Gemiddeld aantal aanwezig en gemiddeld aantal dat er doorheen stroomt zijn dus duidelijk heel verschillend. We zagen een analoog onderscheid zojuist al bij de commando's op de achtergrond. In het hoofdstuk BEURTEN EN KANSEN gaan we op dit soort verschillen nader in.

5.9.3. alternatieve afleidingen

De responsetijdrelaties kunnen in combinatie met de relatie van Little in allerlei gelijkwaardige vormen optreden. Bij deze vormen zijn steeds weer verklarende redeneringen te geven, maar het blijkt dan moeilijker om strikt de gebruikte veronderstellingen te rubriceren. Probeer maar eens na te gaan of er in de volgende verklarende redeneringen moet worden aangenomen dat op alle terminals gelijkwaardig werk wordt gedaan. We weten uit het voorgaande dat dit helemaal niet nodig is.

Zo'n met de responsetijdrelatie gelijkwaardige vorm is

$$\text{gem. aantal in response} = s \times (\text{gem. responsetijd}) / (\text{gem. responsetijd} + \text{gem. denktijd})$$

De redenering hierbij loopt als volgt. Een gebruiker heeft òf een opdracht ingebracht en is in de responsefase òf hij denkt. Hij verblijft gemiddeld gedurende een tijdsduur gem. responsetijd in de responsefase en hij denkt gemiddeld tijdens gem. denktijd. De gebruiker draagt dus met een kans gem. responsetijd/ (gem. responsetijd + gem. denktijd) bij in het aantal in het systeem aanwezige opdrachten. Zijn er s gebruikers dan is het gemiddeld aantal aanwezige opdrachten s maal deze kans.

Een andere redenering merkt op dat per cycle één opdracht wordt ingebracht. Iedere cycle duurt steeds een denktijd en een responsetijd. Per terminal wordt per tijdseenheid dus gemiddeld 1/(gemiddelde cycleduur) opdrachten ingebracht. Aan gemiddeld gem. s terminals dus gemiddeld gem. s / (gem. denktijd + gem. responsetijd) opdrachten. De (gemiddelde) doorstroom is daarom gem. s / (gem. denktijd + gem. responsetijd). Dat is weer een herschreven vorm van de responsetijdrelatie.

De redeneringen zijn nogal intuïtief, maar op zich correct. Ook zo blijkt dat de responsetijdrelaties een verband aangeven dat telkens optreedt in globale beschouwingen rond de response van computersystemen.

5.10. SAMENVATTING

Via de operationele schatters hebben we gezien hoe 'vanzelfsprekend' de responsetijdrelaties zijn. Even natuurlijk als de relatie van Little! Tussen de echte gemiddelde waarden voor responsetijd, denktijd, aantal terminalisten en doorstroom bestaat steeds de relatie

$$\textit{gemiddelde responsetijd} = (\textit{gemiddeld aantal gebruikers})/\textit{doorstroom} - \textit{gemiddelde denktijd}$$

De gemiddelde responsetijd wordt voortaan ook vaak aangegeven met R en de gemiddelde denktijd met Z, de bijbehorende gemiddelde aantallen met r en z, de doorstroom is d. Het voorzetsel *gem.* laten we weg als uit de context duidelijk blijkt dat het (echte) gemiddelde wordt bedoeld. Daarmee is de responsetijdrelatie kortweg

$$R = s/d - Z$$

Alle vier grootheden hebben betrekking op hetzelfde soort of type werk. Zo'n soort of type kan op vele manieren gespecificeerd zijn. Een nader kenmerk van het werk kan zijn dat het via bepaalde terminals binnenkomt. De gemiddelde responsetijd en denktijd zijn dan de gemiddelden aan die terminals, doorstroom is de totale doorstroom van al die terminals samen en gemiddeld aantal terminalisten is het gemiddeld aantal van die terminals. De relatie kan specifiek betrekking hebben op werk dat gebruik maakt van een bepaalde faciliteit; de relevante terminalisten benutten die faciliteit. Het soort werk kan opgebouwd zijn uit een heleboel aparte typen; de relatie slaat ook op de 'overall' waarden.

5.11. OPGAVEN

opgave 5.1: terminalbezetting

Het uitgewerkte voorbeeld 'terminalbezetting' in §5.7 is sterk analoog aan opgave 4.1 uit het hoofdstuk DE RELATIE VAN LITTLE. Vergelijk beide opgaven. Wanneer kan in deze gevallen over tijdens de meetduur ongeveer gelijkblijvende omstandigheden worden gesproken?

opgave 5.2: groepen

Op een computerconfiguratie zijn twee groepen terminalisten aangesloten.

De ene groep van 23 terminalisten brengt werk in met een gemiddelde responsetijd van 0.5 sekonden, hun gemiddelde denktijd is 2 sekonden.

De andere groep van 10 terminalisten brengt gemiddeld 1 opdracht per sekonde in, hun gemiddelde denktijd is 9 sekonden.

- Bepaal de verhouding van de doorstromen aan opdrachten van de beide groepen.
- Hoeveel opdrachten zijn er gemiddeld in de responsefase? Hoeveel zijn er dat van de groep van 23?
- Controleer of de 'overall' gemiddelde responsetijd voldoet aan de 'overall' responsetijdrelatie.

opgave 5.3: vervolg denk/responsetijd (5.3.1, 5.9.1)

Neem aan dat het systeem thans op zijn top werkt, het kan niet meer programma's van het soort, als nu aangeboden, per sekonde verwerken. Toch geeft het systeembeheer de toegang tot de machine vrij, zodat na enige tijd niet 10, maar 20 terminalisten staan ingelogd. Alle terminalisten brengen gelijkwaardige opdrachten de machine in.

- Voorspel met welke gemiddelde responsetijd de terminalist te maken krijgt, die als 20-ste inlogt. Voorspel ook de gemiddelde responsetijd voor zijn buurman, die als 11-de inlogde.
- Geef commentaar op de volgende bewering: Zodra nòg een terminalist inlogt, neemt de responsetijd voor de thans ingelogde terminalisten gemiddeld toe met de inverse van de doorstroom.

opgave 5.4: basisrelatie pollen

Stations delen samen ('sharen') een medium. Dit medium kan steeds door maar hoogstens één van hen benut worden. Daarom moet er een dienstregeling worden onderhouden. De stations krijgen in een bepaalde vaste volgorde de gelegenheid het medium te benutten; zodra de een klaar is, krijgt de volgende de beurt.

Lokale netten waarin een 'token' wordt rondgestuurd werken volgens dit principe (zie bijvoorbeeld de opgave TOKEN-RINGNETWERK uit §13.7). De regeling is het bekendst van het 'pollen' van stations (§7.5.4). Er is bij dit 'samen delen' steeds een belangrijke operationele betrekking werkzaam, die we nu voor het geval van 'pollen' gaan formuleren.

Bij pollen worden de stations cyclisch een voor een gevraagd of ze iets willen verzenden. Als een station informatie heeft, is het gedurende de verzendtijd bezig deze informatie als één info te verzenden. Als er geen informatie is, verzendt het gedurende de nultijd een nulbericht. Na het verzenden van de info of het nulbericht duurt het enige tijd —de overstaptijd— voor het volgende station gevraagd wordt. Op ieder moment wordt er in het systeem een info of een nulbericht verzonden of is het systeem in een overstaptijd. De tijd wordt dus verdeeld in verzendtijden, nultijden en overstaptijden.

- In een meetduur van 10.000 tijdseenheden werd aan 1000 stations gevraagd of ze iets wilden verzenden. Hoeveel overstaptijden zijn er geregistreerd? (geef eventueel grenzen aan voor dit aantal.)
  Van de 1000 stations gaven 200 een nulbericht door en 800 informatie. De stations waren gedurende 6000 t.e. bezig met het verzenden van informatie en 400 t.e. met het verzenden van nulberichten. Definieer de operationele schatter voor de gemiddelde verzendtijd en geef zo mogelijk de gemeten waarde er van.
  Idem voor de gemiddelde nultijd en de gemiddelde overstaptijd.
- Definieer de operationele schatter voor de doorstroom aan info's (de doorstroom). Geef ook weer de waarde.
  Idem voor het aantal nulberichten per tijdseenheid (de nulstroom) en het aantal overstappen per tijdseenheid (de pollstroom).
- Laat zien dat de diverse operationele schatters algemeen voldoen aan

$$\begin{aligned} &\textit{gemiddelde verzendtijd} \times \textit{doorstroom} + \textit{gemiddelde nultijd} \times \textit{nulstroom} \\ &\quad + \textit{gemiddelde overstaptijd} \times \textit{pollstroom} = 1 \end{aligned}$$

  met

$$\textit{pollstroom} = \textit{doorstroom} + \textit{nulstroom}$$

- Druk de pollstroom uit in de gemiddelde verzendtijd, nultijd, overstaptijd en de doorstroom.
- De tussenpoos waarmee een station bij het pollen aan de beurt komt, wordt de cycleduur genoemd. Laat zien dat de gemiddelde cycleduur is

$$\textit{gem. cycleduur} = \frac{\textit{aantal stations} \times (\textit{gem. overstaptijd} + \textit{gem. nultijd})}{1 - \textit{gem. doorstroom} \times (\textit{gem. verzendtijd} - \textit{gem. nultijd})}$$

6

Responsetijd en gebruikers

6.1. INLEIDING

Over de invloed van het gebruikersgedrag op de response van de machine valt veel te zeggen.

We willen de responsetijdrelatie gaan gebruiken om het verband tussen gemiddelde responsetijd en aantal terminalisten uiteen te rafelen en de oorzaken voor een stijging van de responsetijd uit te splitsen. De reaktie van de terminalist op de kwaliteit van de response komt aan de orde.

6.2. RESPONSETIJD EN AANTAL INGELOGDE TERMINALISTEN

Een cruciale vraag, die veel systeembeheerders telkens weer moeten beantwoorden, is hoeveel terminalisten op de huidige machine tegelijk kunnen werken en hoeveel het er zouden kunnen zijn op een andere. Antwoord op zo'n vraag is pas te geven als het verloop van de responsetijd versus het aantal aangesloten terminalisten (het aantal 'concurrent users') bekend is voor terminalisten, die werk inbrengen van het type dat voor de thans gewoonlijk aktieve terminalisten karakteristiek is. Het verloop zal uitgebreid gemeten of berekend moeten worden. Daarna moet worden nagegaan welke responsetijd voor de gebruikers tegen de door hen te betalen kosten acceptabel is, enzovoort. Kortom een behoorlijk fors probleem. We zullen zien dat de simpele responsetijdrelaties al een handreiking bieden.

De responsetijd zal met weinig aangesloten aktieve terminals maar heel weinig variëren met het aantal terminals. Eentje meer of minder doet er niet veel toe, want de gebruikers zitten elkaar ook in het interne van de configuratie niet in de weg; de opdrachten hoeven bij de resources als CPU, disk, drum en geheugen niet op elkaar te wachten. De responsetijd hangt onder die omstandigheden maar

zwakjes af van het aantal terminalisten. De (gemiddelde) doorstroom zal evenredig met het aantal terminalisten toenemen.

Alles ligt heel anders als er zeer veel terminals zijn aangesloten en ingelogd. Nu heeft men veel last van elkaar. De responsetijd varieert sterk met het aantal terminalisten. De computer doet zijn best, maar meer dan in een sekonde gemiddeld d_{max} opdrachten —van het door de terminalisten aangeboden soort werk— zal hij niet kunnen halen. De doorstroom hangt onder deze omstandigheden vrijwel niet af van het aantal terminalisten, de doorstroom zit tegen de maximale waarde d_{max} aan. Deze overwegingen brengen we nu in de responsetijdrelatie in.

De gemiddelde responsetijd als funktie van het aantal ingelogde terminalisten s geven we aan als R_s. De gemiddelde denktijd Z_s zal heel weinig afhangen van het aantal terminalisten, ze zien elkaar misschien niet eens; daarom nemen we Z_s eenvoudig constant: Z. Bij een geringe interaktieve werklast is de responsetijd simpel de som van de verwerkduren bij de diverse devices, bij weinig terminalisten zitten de opdrachten elkaar niet in de weg. Zolang er bij de devices niet gewacht wordt is daarom de gemiddelde responsetijd onafhankelijk van het aantal terminalisten: de gemiddelde responsetijd R_s is gelijk aan de gemiddelde responsetijd R_1 bij één ingelogde terminalist. Voor weinig terminalisten is

$$R_s \approx R_1$$

De responsetijdrelatie zegt dan dat voor de doorstroom d_s geldt

$$R_1 \approx s / d_s - Z$$

of

$$d_s \approx s / (R_1 + Z)$$

De doorstroom d_1 bij 1 terminalist is natuurlijk de inverse van de cycleduur $1/(R_1 + Z)$. Dus

$$d_s \approx s\, d_1$$

Bij weinig terminalisten is de (gemiddelde) doorstroom netjes evenredig met het aantal aktieve terminalisten! Bij deze aantallen gelden de voordelen van multiprogrammering. Doordat de devices meer parallel gaan werken bij meer klandizie —ideaal is dat alle devices zoveel mogelijk aktief zijn— kan de totale prestatie van de configuratie als geheel —de doorstroom— toenemen, zonder dat zo'n 'samen delen' voor de individuele benutters verlies aan response betekent.

Maar bij een zware interaktieve werklast, bij veel terminalisten, ligt de zaak

anders. Dan geldt

$$d_s \approx d_{\max}$$

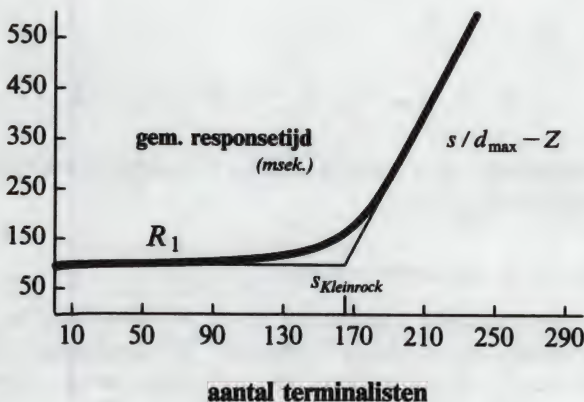
en volgens de responsetijdrelatie

$$R_s \approx s / d_{\max} - Z$$

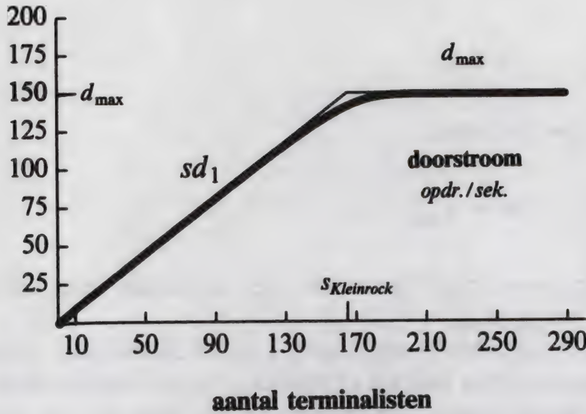
Hier neemt dus de responsetijd lineair toe met het aantal terminalisten, met een richtingscoëfficiënt van $1 / d_{\max}$.

Het verloop van de gemiddelde responsetijd, en de doorstroom, versus het aantal ingelogde terminalisten is dus heel karakteristiek. De gemiddelde responsetijd loopt eerst —bij kleine aantallen ingelogde terminalisten— vlak ($R_s \approx R_1$), terwijl uiteindelijk voor hoge waarden voor het aantal terminalisten de gemiddelde responsetijd lineair toeneemt met het aantal terminalisten ($R_s \approx s / d_{\max} - Z$). Het verloop van de doorstroom is eerst lineair toenemend bij kleine aantallen terminalisten ($d_s \approx s d_1$) en voor hoge waarden van s is het vlak ($d_s \approx d_{\max}$). Deze trends ziet men in de praktijk altijd optreden, ze liggen dwingend besloten in de responsetijdrelaties.

Het algemene verloop van de gemiddelde responsetijd en de doorstroom is geschetst in de figuren 6.1-2. De getrokken lijnen geven het gedrag volgens de besproken —nogal brute— benaderingen; het 'echte' waargenomen gedrag zal vaak zijn volgens de vette lijnen.



Figuur 6.1. Gemiddelde responsetijd als funktie van het aantal terminalisten. De getrokken lijn is de benadering met de responsetijdrelatie; de vette lijn schetst het echte verloop. Het Kleinrockpunt $s_{Kleinrock}$ is het snijpunt van de horizontale asymptoot R_1 en de schuine asymptoot $s / d_{\max} - Z$ (§6.2.1).



Figuur 6.2. (Gemiddelde) doorstroom als funktie van het aantal terminalisten. De getrokken lijn is de benadering met de responsetijdrelatie; de vette lijn schetst het echte verloop. Het Kleinrockpunt $s_{Kleinrock}$ is het snijpunt van de benaderingen sd_1 en d_{max}.

Hellingshoeken en hoogten variëren met de omstandigheden, maar de grondvorm voor de curven is universeel.

Als de doorstroom d_s voor grote waarden van s erg langzaam naar d_{max} gaat, zal er een ander asymptotisch gedrag optreden. Wanneer bijvoorbeeld $1/d_s$ asymptotisch naar $1/d_{max}$ zou gaan als

$$1/d_s \approx 1/d_{max} + 2/s$$

gold

$$R_s \approx s(1/d_{max} + 2/s) - Z \approx s/d_{max} + 2 - Z$$

De gemiddelde responsetijd voor grote aantallen ingelogde terminalisten ligt dan nog hoger dan volgens $s/d_{max} - Z$.

6.2.1. voorbeeld: weinig en veel terminalisten

Op een groot computersysteem zijn thans maar 33 terminalisten aktief. Hun responsetijd is 100 msek., hun denktijd 1 sekonde. De configuratie kan maximaal per sekonde (gemiddeld) 150 opdrachten, als door deze terminalisten ingebracht, verwerken. Op superdrukke dagen zijn er in de piekuren 220 terminalisten aktief.

Bij 33 terminalisten zal het op dit systeem waarschijnlijk niet erg druk zijn. Laten we aannemen dat de gemiddelde responsetijd nog maar weinig verschilt van die bij één werkende terminalist.

De huidige doorstroom is dan gemiddeld:

$$d_{33} = 33 \times d_1 = 33/(0.1+1) = 30 \text{ opdrachten/sek.}$$

Dat is veel lager dan de maximale doorstroom van 150 opdrachten per sekonde.

Bij topdrukte ligt de zaak anders. Dan is het systeem buiten het gebied waarin de doorstroom lineair toeneemt met het aantal terminalisten. Doortrekken van het lineaire verband zou geven:

$$d_{220} = 220/1.1 = 200 \text{ opdrachten/sek.}$$

en dat is ver boven het maximum.

Volgens de responsetijdrelatie is de gemiddelde responsetijd bij 220 terminalisten, omdat de doorstroom kleiner dan 150 is,

$$R_{220} \geqslant 220/150 - 1 = 0.46 \text{ sek.}$$

De gemiddelde responsetijd is bij topdrukte dus minstens 460 msek. in plaats van 100 msek. De gemiddelde responsetijd zal bij toenemend aantal terminalisten eerst lange tijd ongeveer 100 msek. zijn en daarna toenemen, tot een lineaire stijging volgens $s/150 - 1$.

Een globale schatting voor het 'afbuigpunt' in de grafiek van de gemiddelde responsetijd (en van de doorstroom), —dus het aantal terminalisten waarbij de response duidelijk begint te stijgen— wordt gegeven door het *Kleinrockpunt*. LEONARD KLEINROCK, een van de pioniers van de prestatie-analyse van computernetwerken, stelde indertijd een eigenlijk voor de hand liggende afschatting voor. De afschatting is bekend geworden als 'Kleinrockpunt'. Het zal wel niet verbazen dat het Kleinrockpunt ligt bij een waarde van s, waarvoor

$$d_{max} = s\, d_1$$

KLEINROCK vond dat er eigenlijk niet veel meer dan $s_{Kleinrock} = d_{max}/d_1 = 150 \times 1.1 = 165$ terminals ingelogd kunnen zijn.

In modellen van computerconfiguraties kan worden nagegaan, hoe goed het Kleinrockpunt aangeeft waar het afbuigen begint. Voor de moderne mainframes zal de gemiddelde responsetijd tot aan het afbuigpunt veel korter zijn dan de gemiddelde denktijd, speciaal zal d_1 vrijwel $1/Z$ zijn. Het Kleinrockpunt ligt dan bij

$$s_{Kleinrock} = Z\, d_{max}$$

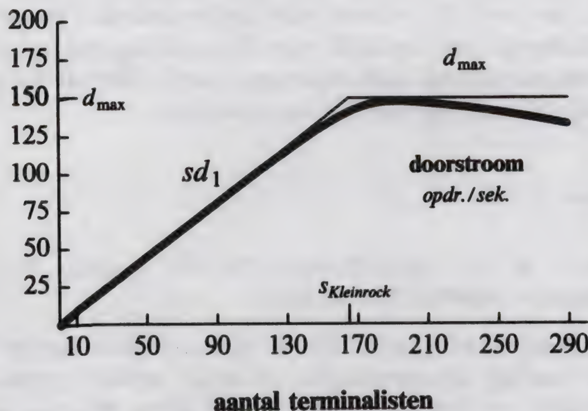
6.2.2. *inzichten responsetijd versus aantal terminalisten*

Vooral opvallend is de lineaire stijging van de responsetijd als het druk is (we vereenzelvigen even drukte en groot aantal ingelogde terminalisten). De last van de vele terminalisten kan zelfs, als de ontwerpers van het operatingsysteem niet hebben opgepast of als de systeembeheerder niet oplet, nog drukkender zijn dan volgens deze lineaire stijging. In dat geval moet de processor ook *relatief* meer boekhouding doen naarmate het drukker wordt (gebruikers inswappen, helpen bij pagineren, beheren wachtrijen, enzovoort), waardoor zijn doorstroom achteruit loopt: hij houdt maar steeds minder tijd over voor nuttig werk. Het is normaal dat er per opdracht wat 'overhead' wordt gepleegd, dat is systeemwerk ten nutte van allen. Maar wat de processor nu per opdracht aan werk moet presteren blijft niet gelijk, maar groeit met de drukte! Men zegt dan kortweg: de overhead neemt sterk toe en bedoelt dat het werk per opdracht toeneemt. Ook de randapparaten kunnen per opdracht meer werk te verteren krijgen door extra paging en swapping.

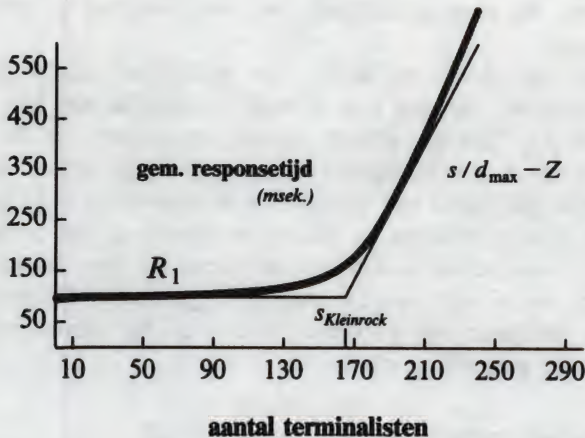
Een tekenend verloop voor de doorstroom en de responsetijd voor deze gevallen staat in de figuren 6.3-4.

Het verloop van d_s voor grote s is dan niet $d_s \approx d_{\max}$ maar d_s wordt dalend. De responsetijdrelatie leert dan dat de gemiddelde responsetijd voor grote drukte meer dan lineair toeneemt; als de doorstroom aanzienlijk terugloopt zal de responsetijd een haast exponentiële toename vertonen.

Merk wel op dat het oplopen van de response al optreedt als dit verschijnsel zich helemaal niet voordoet, zoals we in de voorgaande paragraaf in de figuren 6.1 en 6.2 zagen.



Figuur 6.3. (Gemiddelde) doorstroom als funktie van het aantal terminalisten. Overhead per opdracht neemt toe met de drukte. De getrokken lijn is de benadering met de responsetijdrelatie; de vette lijn schetst het echte verloop.



Figuur 6.4. Gemiddelde responsetijd als funktie van het aantal terminalisten. Overhead per opdracht neemt toe met de drukte. De getrokken lijn is de benadering met de responsetijdrelatie; de vette lijn schetst het echte verloop.

Heel vaak wordt gesuggereerd dat in het algemeen een toename van de responsetijd zou worden veroorzaakt door extra 'overhead'. Dat is dus zeker niet waar, ook als extra overhead vermeden kan worden stijgt de responsetijd als het drukker wordt.

Een overdadig optreden van page faults heet thrashing, dit veroorzaakt veel extra overhead. Het in elkaar klappen van de doorstroom door dit thrashen was in het verleden de schrik van de beheerders van systemen met gepagineerd virtueel geheugen. In de gedistribueerde verwerking in netwerken kan hetzelfde fenomeen ontstaan door lokale deadlock en onbeperkte retry.

Wanneer de doorstroom niet afneemt met toenemende s (als in figuur 6.1-2) doet het systeem zich aan de gebruikers bij grote toenemende drukte als steeds trager voor, maar wordt het niet trager. Alle oponthoud ontstaat enkel door het moeten 'sharen' van een beperkte verwerkingssnelheid met steeds meer gebruikers. Maar als de doorstroom afneemt voorbij de maximale doorstroom (als in figuur 6.3-4) lijkt het systeem niet alleen trager, maar is het ook trager.

Het valt tegen om buiten de responsetijdrelaties om door vlotweg redeneren even te laten zien dat de responsetijd bij grote drukte evenredig met het aantal terminalisten toeneemt. Met de responsetijdrelaties ging het 'vanzelf', in het voorbeeld uit het vorige hoofdstuk over de responsetijdrelaties in een systeem met een beperkte capaciteit (§5.5.2) is de evenredige toename al direkt gedemonstreerd.

Het inloggen van nòg een terminalist maakt bij grote drukte dat de gemiddelde responsetijd van alle terminalisten stijgt van $R_s = s/d_{max} - Z$ tot $R_{s+1} = (s+1)/d_{max} - Z$. De gemiddelde responsetijd voor de reeds ingelogde terminalisten neemt dus toe met $1/d_{max}$. Dit is de tijd, die de configuratie gemiddeld per opdracht bezig is. Het is dus net alsof (inderdaad: alsof) alle 'oude'

terminalisten de opdracht van de nieuwe voor moeten laten gaan. Maar die profiteert er niet van, hij moet gemiddeld even lang wachten op response als zij. Wat is er aan de hand?

Van belang voor het doorzicht is dit: het gemiddeld aantal denkende terminalisten neemt door het inloggen van de extra terminalist niet toe. Er waren er $Z\,d_{max}$ en dat blijft zo. Het gemiddelde aantal terminalisten in de responsefase neemt dus met één toe na het inloggen van een terminalist extra. Een terminalist extra betekent simpel gemiddeld een 'terminalist in response' extra. De relatie van Little toegepast op alle opdrachten in de responsefase zegt dat gemiddeld ieders verblijfduur in de responsefase daardoor ook toeneemt met de inverse van de gemiddelde stroom, die al zijn hoogste waarde d_{max} heeft. De oorzaak van de lineaire stijging is blijkbaar het gefixeerd raken van het gemiddeld aantal terminalisten dat niet om verwerking vraagt.

6.2.3. vervolg voorbeeld: weinig en veel terminalisten (6.2.1)

Bij 33 terminalisten denken er gemiddeld (Little)

$$\text{gem. denkend} = \text{doorstroom} \times \text{gem. denktijd} = 30 \times 1 = 30 \text{ terminalisten}$$

En er zijn gemiddeld in de responsefase

$$\text{gem. in response} = 30 \times 0.1 = 3 \text{ terminalisten}$$

Ons uitgangspunt was dat het bij 33 terminalisten niet druk is. Dit houdt kennelijk in dat de gemiddeld 3 opdrachten in het systeem elkaar niet in de weg zitten.

Bij 220 terminalisten zullen er gemiddeld hoogstens denken

$$\text{gem. denkend} \leqslant 150 \times 1 = 150 \text{ terminalisten}$$

zodat er dus minimaal in de responsefase zijn

$$\text{gem. in response} \geqslant 150 \times (220/150 - 1) = 70 \text{ terminalisten}$$

De 70 opdrachten, die het computersysteem bevolken als er 220 terminalisten werken, zullen elkaar danig in de weg zitten!

6.2.4. configuratie als server (wachttijdtheoretisch intermezzo)

We hebben bij het bespreken van wat er gebeurt als de configuratie volloopt, alleen de responsetijdrelatie gebruikt; de hele redenering was daarmee gebaseerd op de relatie van Little. De configuratie kan algemeen gezien worden als een server in een wachttijdensysteem met de opdrachten van de terminalisten als klanten. Wat zouden enkele 'voor de hand liggende' aannamen uit de wachttijdentheorie hier zeggen? (vergelijk later het hoofdstuk INLEIDING WACHTTIJDENSYSTEMEN.)

De server 'computerconfiguratie' krijgt bij drukte een vaste verwerkingssnelheid, over één opdracht doet hij gemiddeld $1/d_{max}$ t.e. Iedere ingebrachte opdracht moet dus in elk geval rekenen op $1/d_{max}$ t.e. aan verwerking. De opdracht zal bovendien opgehouden worden doordat hij de verwerkingscapaciteit moet delen, moet 'sharen': er dingen gemiddeld $\bar{r}$ opdrachten naar de hulp van de server. Als de afwikkelingsvolgorde FCFS is, de verdeling van de serviceduren negatief exponentieel (§12.3.6) en een opdracht bij lukraak inbrengen gemiddeld $\bar{r}$ opdrachten 'in response' aantreft, is de bijdrage aan de responsetijd door het wachten op anderen gemiddeld $\bar{r}/d_{max}$. De gemiddelde responsetijd zou daarmee zijn

$$\bar{R} = (\bar{r}+1)/d_{max}$$

De afwikkelingsvolgorde is in werkelijkheid natuurlijk helemaal niet FCFS —dat zou bizar zijn— en over de verdeling van de serviceduren weten we niets, maar misschien zou men graag geloven dat het resultaat in het algemeen ongeveer correct is. De responsetijdrelatie voor grote drukte zegt echter dat het fout is, volgens Little (§5.9) is vrijwel

$$\bar{R} = \bar{r}/d_{max}$$

We staan hier niet langer bij stil, de beschrijving is erg onnauwkeurig; een betere analyse is mogelijk via de 'aankomststelling' voor separabele netwerken (§14.9.2, het gaat om een gesloten systeem). Duidelijk is dat de relatie van Little even rijk is als een 'wachttijdentheorie' op dit niveau, ja deze zelfs corrigeert. Merk op dat de redeneringen via Little voor elke afwikkelingsregeling (scheduling) gelden; de responsetijdrelaties zijn onafhankelijk van de scheduling.

6.2.5. voorbeeld: commando's op de achtergrond

Een terminalist vindt het handig om af en toe een opdracht 'op de achtergrond' te laten lopen. Van zo'n opdracht neemt hij pas weer nota als hij de resultaten nodig heeft. Ondertussen kan hij gewoon doorgaan met zijn interaktieve besognes. Hij doet —zo denkt hij— meer nuttig werk, omdat hij bezig blijft. Welke consequenties heeft deze handelwijze voor het algemeen belang? De responsetijdrelatie zegt er iets over.

Het door de terminalisten ingebrachte werk wordt gesplitst in de soorten 'achtergrond' en 'interaktief'. Laten we aannemen dat de gebruikers nogal veel gebruik maken van de extra faciliteit: op elke 4 interaktieve opdrachten starten ze een opdracht op de achtergrond. De overall doorstroom bestaat uit (vergelijk de discussie over soorten gebruikers uit het vorige hoofdstuk, §5.6.1)

$$d_{overall} = d_{achtergrond} + d_{interaktief}$$

met $d_{interaktief} = 4/5\, d_{overall}$. De tijd nodig voor het opbouwen van een achtergrondopdracht zal volledig geteld moeten worden als denktijd voor de volgende interaktieve opdracht. Als er geen extra aandacht nodig is voor het goede verloop van het werk, zal de gemiddelde denktijd bij het interaktieve werk een factor 5/4 groter zijn dan wanneer er niets op de achtergrond zou draaien (kijk naar de operationele schatter 'Z). Als het druk is en er niets op de achtergrond zou draaien, gold vrijwel

$$R = s / d_{\max} - Z$$

Met achtergrondcommando's is echter dan vrijwel $d_{interaktief} = 4/5\, d_{\max}$ (als de overhead per opdracht niet toeneemt door de achtergrondcommando's) en

$$R = s / (4/5\ d_{\max}) - 5/4\ Z \quad = 5/4\ (s / d_{\max} - Z)$$

Door de commando's op de achtergrond doet de machine zich aan de interaktieve gebruikers als trager voor, hun gemiddelde responsetijd wordt 'opgeblazen' met de factor 5/4. We zien hier, net als bijvoorbeeld in §5.5.2, dat de beperkte verwerkingssnelheid van de computerconfiguratie maakt dat het gemak van de achtergrondcommando's moet worden bekocht met een groeiende responsetijd.

6.3. PRODUKTIVITEIT VAN DE COMPUTERGEBRUIKER

In het voorgaande hebben we af en toe —impliciet of expliciet— aangenomen dat de aangeboden werklast niet varieert met de werkomstandigheden tijdens het aanbieden. Het soort werk, door de gebruikers aangeboden, en de gemiddelde denktijd waarmee ze dit aanbieden, zouden niet door de omstandigheden worden beïnvloed. Deze veronderstelling werd gemaakt om de hoofdlijnen van de redenering zuiver naar voren te laten komen, om de hoofdeffekten af te zonderen. Veel gebruikers zullen voor zichzelf het gevoel hebben dat ze af en toe danig reageren op de prestaties van de computer. Het is daarom misschien goed wat nader in te gaan op de koppeling tussen deze meester —de computer?— en zijn slaaf —de gebruiker?— en te kijken welke invloeden hier een rol spelen als neveneffekt.

Het is natuurlijk zaak duidelijk onderscheid te maken tussen het reageren op calamiteiten, waarbij de responsetijd tijdelijk drastisch verandert, en het inspelen op een blijvende verandering in de gemiddelde responsetijd. Als er een hardwarestoring in de lucht zit, of bij een zich plotseling manifesterende systeembug, is het al heel gauw 'sauve qui peut'. Er worden vlot red-akties ondernomen en er wordt gepoogd er zo snel mogelijk —via de machine!— achter te komen wat er aan de hand is. De overvloed aan opvraagboodschappen van de bezorgde gebruikers zal het systeem stellig nog verder de mist in helpen. Berucht zijn de lawines aan systemcalls om de inhouden van jobwachtrijen te laten zien.

Maar dit zullen niet de effekten zijn die bedoeld worden als de wisselwerking tussen responsetijd, werklast en denktijd ter discussie wordt gesteld. Het zijn —hopelijk— geen beklijvende veranderingen in de werkomstandigheden.

Bij een blijvende verslechtering van de response zal de gebruiker er van uitgaan dat de responsetijd gemiddeld is, zoals die is; hij gaat er mee leven. Zijn werk zal hij moeten klaren en daarbij heeft hij, zo goed en zo kwaad als het gaat, de computer nodig. Als iemand geconcentreerd bezig is met zijn werk zal hij voor het uitvoeren van de komende handelingen gebruik maken van zijn 'korte termijn geheugen'. In dat deel van zijn hersenreservoir heeft hij de 7 (ongeveer 7: magisch getal) characters, woorden en icons opgeslagen, die hij aanstonds zal intikken, de komende reakties die hij zal geven, enzovoort. Daarnaast zal hij een 'korte termijn plan' in zijn hoofd hebben met de beslissingen, die hij zal nemen na de aanstaande reakties van zijn werkgenoot de computer en de gronden waarop hij tot die beslissingen zal komen.

Beide, korte-termijn-geheugen en korte-termijn-plan, zullen verstoord raken als hij bij zijn werk wordt afgeleid door wat er om hem heen gebeurt. Maar ook zijn hulp de computer kan voor die verstoring zorgen. Als de response te lang uitblijft verdampt de inhoud van zijn korte-termijn-geheugen. Hij gaat fouten maken en door de irritatie daarover raakt ook zijn korte-termijn-plan in de war, temeer omdat hij in de tijd, die hij nu over heeft, aan heel andere zaken dan zijn direkte werk gaat denken. Als de response te snel komt meent de gebruiker dat hij zich moet aanpassen aan de snelheid van de computer, hij wordt bang de computer niet te kunnen bijbenen. Ook dan is het resultaat dat hij fouten gaat maken en verkrampt raakt, waaronder weer zijn korte-termijn-plan lijdt. Het lezen van schermteksten bij een zichtbaar te snel displayen is vermoeiend, de blik kan de cursor niet volgen.

In de loop van de jaren is er heel wat onderzoek gedaan naar de orde van grootte van deze effekten. Er zijn nog altijd betrekkelijk weinig tastbare resultaten, maar het is inmiddels wel duidelijk geworden dat de mate waarin deze effekten optreden nogal afhangt van de aard van het werk, het maakt veel uit of het routinewerk is of werk waarvoor telkens een stukje denkwerk moet worden gepleegd, zoals bij CAD (Computer Aided Design) of programmeerwerk. Natuurlijk is een zeer snelle response altijd heel prettig. Een responsetijd van minder dan een sekonde komt meestal over als 'onmiddellijk'. De computer is dan zo snel dat de dwang hem bij te houden verdwenen is en er rustig en ontspannen kan worden gewerkt. Het

lezen van schermteksten kan nu op de normale manier gebeuren omdat een hinderlijk verschil tussen de leessnelheid en de afdruksnelheid niet optreedt: de te lezen tekst is steeds in zijn geheel aanwezig. Het enige nadeel van een snelle response —naast de kosten: het is wellicht luxe— is eigenlijk alleen maar dat de gemakzucht toeneemt, waardoor er verkeerde beslissingen worden genomen, die hersteld moeten worden. Reparatie van kleine fouten is heel snel mogelijk; wat meer foutjes bij een snelle response kan zelfs nog voordeliger uitkomen dan minder foutjes bij een langzamer response. Er is echter geconstateerd dat de gebruikers bij wat langere responsetijden betere wegen bewandelen om hun doel te bereiken en met minder commando's toe kunnen.

Voor korte opdrachten, die weinig van de machine vergen, is eigenlijk een responsetijd van minder dan 2 à 3 sekonden een eis. Krachtiger opdrachten mogen er wat langer overdoen voordat er hinder wordt ondervonden. De verwachtingen van de gebruiker spelen in de hele interaktie een belangrijke rol, de gebruiker voelt zich happy met de response die hij redelijk vindt op grond van zijn ervaring en is blij met een incidentele korte response. De verwachting van velen dat de gebruiker een geringe spreiding in de response erg waardeert, is niet overtuigend bevestigd. Soms worden responsetijden gelijk getrokken door het inbouwen van onzinnige delays voor al te korte responsen, dit lijkt dus de gemiddelde gebruiker geen goed te doen.

De manier waarop de gebruiker zijn werk verricht wordt dus inderdaad beïnvloed door de responsetijden, die hij gemiddeld ondervindt. In een stationaire fase van verwerking blijken de aangegeven verschijnselen echter duidelijk bijeffekten te zijn. De afhankelijkheid zal in een nadere, toegespitste, probleemstelling goed als 'hogere orde' effekt kunnen worden ingebouwd.

Wat is de produktiviteit van de gebruiker die hieruit volgt? In eerste instantie (in eerste orde) zal de samenstelling van de werklast niet veranderen als de responsetijd daalt door het invoeren van betere faciliteiten en de gemiddelde denktijd verandert ook niet. De responsetijdrelatie leert dan simpelweg dat door de daling van de gemiddelde responsetijd de doorstroom toeneemt: de eenvoudige overweging dat er dan meer kan gebeuren in dezelfde tijd gaat echt op! Betere faciliteiten in de zin van lagere gemiddelde responsetijden maken zo dat er per werkuur meer werk kan worden verricht en er dus economisch gezien efficiënter gewerkt wordt (dat zagen we al in §5.5.2). De produktiviteit van de werknemer neemt toe.

We wagen het er op te wijzen dat de hierbij vaak gememoreerde belangentegenstelling tussen management en gebruikers soms op spraakverwarring berust. De gebruiker zou belang hebben bij een snelle response, de manager bij een hoge doorstroom en die belangen —zo zegt men— moeten wel botsen. Zojuist zagen we dat het management ook belang heeft bij een korte response, want een korte response gaat samen met een hoge doorstroom en dus wordt er bij een korte response per uur veel werk verricht. Paradox? Nee. In de bekende uitspraak wordt met een hoge doorstroom bedoeld een verwerking met relatief weinig 'overhead'! Of nog iets preciezer gezegd: een verwerking met een zo hoog mogelijke bezettingsgraad van de CPU door gebruikerprogrammatuur en zo weinig mogelijk

door 'systeemprogrammatuur'. De verwerking van een interaktieve werklast betekent altijd veel 'regelwerk' voor de CPU, dit in tegenstelling tot bijvoorbeeld een eenvoudiger te besturen batch-verwerking. De uitspraak zegt dus dat het management geneigd is om het interaktieve werk terug te dringen omdat het zo duur aan computer-power is, het kost zoveel overhead. Terugdringen betekent het toekennen van een geringer deel van de CPU-tijd, waardoor de machine voor het interaktieve werk effektief trager wordt en de responsetijd dus oploopt. De uitspraak poneert dus een stelling op het gebied van de optimale samenstelling van de werklast, relatief veel interaktief werk zou ongunstig zijn. Over de intrigerende problemen die hier liggen valt veel te zeggen, maar we vrezen nu al dat de uitspraak —gezien ook de verwarrende formulering— wel eens aan de oppervlakkige kant zou kunnen zijn. Het is een van de weinige plaatsen waar het in de wandelgangen gangbare spraakgebruik met doorstroom ineens niet een grootheid op gebruikersniveau bedoelt, maar het primitieve concept van 'doorstroom = aantal user-CPU instrukties per tijdseenheid' hanteert om een lagere doorstroom aan interaktieve programma's af te wegen tegen bijvoorbeeld een hogere batchstroom (vergelijk de vaagheden rond de mipsrate in §2.2.1).

Met de uitspraak kan soms ook simpelweg bedoeld worden dat de responsetijd onevenredig stijgt als de doorstroom te ver wordt opgevoerd door forse belastingen, zeg als deze boven het Kleinrockpunt komen. Deze situatie hebben we al bekeken. De individuele doorstroom, en dus de individuele produktiviteit, loopt achteruit. De loonkosten van extra terminalisten zijn spoedig weggegooid geld.

6.4. NABESCHOUWING

Hoewel de responsetijdrelaties erg algemeen zijn en dus betrekkelijk weinig informatie-inhoud hebben —in wezen zijn het relaties van Little— bleken ze toch erg nuttig te zijn om tot correcte concrete redeneringen te komen.

Hun praktisch toepassingsgebied reikt ver, verder dan de tot nu toe op de voorgrond getreden context. In de voorgaande beschouwingen over responsetijd versus aantal terminalisten lieten we het een beetje in het midden of er nog ander werk dan dat van de terminalisten op de machine draait. Dit was ook niet nodig: de responsetijdrelaties gelden ook per groep en de beschouwingen gelden onverkort als het computersysteem diverse werklasten verwerkt, waarvan de door de betrokken terminalisten interaktief ingebrachte last er maar één is. Er zouden nog heel andere interaktieve groepen bezig kunnen zijn, er kan een batchstroom lopen, enzovoort. Grenzen aan de doorstroom worden dan mede gesteld door het moeten verwerken van de 'overige' werklast. Maar gegeven het verloop van de doorstroom volgt weer het verloop van de responsetijd.

In moderne configuraties komt veel werk binnen via netwerklijnen, de terminalisten zitten dan op totaal verschillende locaties. Een aanzienlijk deel van de responsetijd kan dan veroorzaakt worden door de transfertijd over de lijn, compleet met de vertragingen door wachten die daarbij optreden. De systeembeheerders van de host-configuratie hebben geen greep op deze vertragingen, deze vallen buiten hun beheersgebied en dat kan aanleiding geven tot heel wat

frustraties over en weer. Maar gelukkig gelden ook dan de responsetijdrelaties.

De relaties gelden strikt ook voor de gemiddelde totale overdrachttijd —inclusief wachten— bij berichtenverkeer over een door meerdere gebruikers benut ('gesha-red') communicatiemedium. Zo'n medium kan de coaxkabel zijn in een lokaal netwerk, of een bus, of een telefoonlijn. Bij de analyse van de prestaties van protocollen voor deze communicatie blijken de responsetijdrelaties exact te gelden, bijvoorbeeld in 'slotted Ethernet' voldoet de analytische oplossing aan deze relatie.

De responsetijdrelaties treden altijd op als een beperkt aantal 'gebruikers' parallel in de tijd een beperkt aantal 'resources' sharen. Bij de analyse van computerconfiguraties en -netwerken moet altijd onderscheid worden gemaakt tussen *open* en *gesloten* (deel)systemen (§14.8). In gesloten systemen maken de bronnen van de opdrachten en de servers, die de opdrachten verwerken, deel uit van eenzelfde groot verwerkingssysteem; in die systemen komen geen opdrachten van 'buiten', van 'elders', het systeem binnen. Dat is wel het geval in wat 'open' systemen genoemd worden. Heel in het algemeen gesproken komen onze responsetijdrelaties in gesloten (deel)systemen altijd naar voren.

6.5. OPGAVEN

opgave 6.1: te simpel?

Op een computersysteem zijn een groot aantal terminals aangesloten, waarvan de gebruikers alle hetzelfde type opdrachten aanbieden. De gemiddelde denktijd bij dit werk is 5 sekonden per opdracht. Het computersysteem werkt op de maximale capaciteit, er verlaten per sekonde gemiddeld 4 afgewerkte opdrachten het systeem. Een van de 50 terminalisten schat zijn gemiddelde responsetijd op de volgende simpele manier:

> De machine kan niet meer dan 4 opdrachten per sekonde verwerken. Hij is dus per opdracht 1/4 sekonde bezig.
>
> Een komende opdracht van mij moet worden uitgevoerd, maar zal ook moeten wachten op opdrachten van andere gebruikers. De opdracht zal gemiddeld de helft van de opdrachten van de andere gebruikers voor moeten laten gaan.

Mijn responsetijd zal dus gemiddeld bestaan uit $1/4 + 49/2 \times 1/4 = 51/8$ sekonden. Geef commentaar op deze berekening. Eens of oneens? En waarom?

opgave 6.2: extra gebruiker

Op een computersysteem werken, onder andere, terminalisten die met een gemiddelde denktijd van 2 sekonden interaktieve opdrachten inbrengen. Er zijn momenteel 400 terminalisten aktief, het systeem verwerkt per sekonde 160 van hun opdrachten.
Op een zeker moment logt er nòg een gebruiker in. Deze gebruiker werkt heel efficiënt, hij heeft bijvoorbeeld een gedeelte van zijn opdrachten op een meermalen

door de commando-interpretator te lezen file gezet. De gebruiker blijkt een doorstroom te halen van 2 opdrachten per sekonde bij een responsetijd van 200 msek.

- Wat is zijn gemiddelde denktijd?
- Wat is tijdens het meedoen van deze extra gebruiker de gemiddelde responsetijd van de 400 'gewone' terminalisten?

Neem eerst het geval dat het op het systeem ook met de extra gebruiker niet druk is. Daarna dat het zonder de extra gebruiker al op zijn top werkt.

opgave 6.3: vollopen configuratie

Een groot computersysteem biedt aan een bepaald type gebruikers een beperkte service. Gebruikers van dit type brengen hun opdrachten interaktief in met een gemiddelde denktijd van 1 sekonde. De configuratie zal maximaal gemiddeld 145 van deze opdrachten per sekonde verwerken. Normaliter, als er zo'n 10 terminalisten van dit type zijn ingelogd, is hun gemiddelde responsetijd 100 msek. Op superdrukke dagen zijn er in de piekuren echter 225 terminalisten aktief.

- Maak een afschatting voor de gemiddelde responsetijd bij 220 ingelogde terminalisten.

Iemand denkt dat de maximale doorstroom bij 140 terminalisten al vrijwel bereikt is. Hij bepaalt de gemiddelde responsetijd bij 140 terminalisten op grond van deze aanname en concludeert dat de responsetijdrelatie niet opgaat!

- Geef aan wat de fout is in zijn redenering.

(negatieve gemiddelde responsetijd wijst op foute waarde voor gemiddelde s, denktijd of doorstroom.)

opgave 6.4: vaste snelheid configuratie

Een klein computersysteem bestaat uit eigenlijk maar één server (het merendeel van de devices in het systeem is relatief snel vergeleken met deze server). Deze server heeft een vaste verwerkingssnelheid, het opvoeren van de multiprogrammeringsgraad maakt hem niet sneller. Dit systeem wordt benut door gebruikers met een gemiddelde denktijd van 5 sekonden. Wanneer er maar één gebruiker op het systeem werkt, is de gemiddelde responsetijd 1 sekonde.

- Laat met behulp van de relatie van Little zien dat de doorstroom niet hoger kan zijn dan 1 opdracht per sekonde.
- Bepaal de doorstroom van een gebruiker die alleen op het systeem werkt. Hoeveel gebruikers kunnen er hoogstens met een redelijke response tegelijk op het systeem werken (Kleinrockpunt)? Schets de grafiek van de responsetijd en de doorstroom als funktie van het aantal gebruikers ($0 \leqslant$ aantal gebruikers $\leqslant 10$).

- Het systeem krijgt een 'upgrade'. Een gebruiker heeft daarna, als hij alleen op het systeem werkt, een gemiddelde responsetijd van 0.8 sekonden. De maximale doorstroom van het systeem is nu 1.25 opdrachten per sekonde. Schets opnieuw de grafiek van de responsetijd als funktie van het aantal gebruikers.
- Nuanceer de uitspraak: 'Het systeem is voor de gebruiker 20% sneller geworden'.

opgave 6.5: Client/Server

Een lokaal netwerk verbindt werkstations via een snelle (ETHERNET) verbinding. Op de werkstations zijn terminals en disks aangesloten. Gebruikerprogramma's plegen acces op zowel hun lokale randapparaten als —via het net— op disks elders.

Een gebruiker kan van zo'n remote-disk lezen en erop schrijven doordat in het lokale operatingsysteem en in de netwerksoftware een Client/Server-faciliteit is ingebouwd. Op zijn werkstation bevindt zich een Clientsystem, dat de 'remote-I/O'-opdrachten op de normale wijze als I/O-opdracht afwikkelt, maar de fysieke operaties doorstuurt naar het station, waaraan de betrokken remote-disk hangt. Op dit station, de host, worden deze opdrachten verwerkt door een Serversysteem, dat de opdrachten als fysieke operatie doorgeeft aan de bedoelde disk, het betrokken Clientsysteem op de hoogte stelt van de afloop en informatie overdraagt.

De verwerking van een I/O-opdracht op een remote-disk kost aan CPU-tijd op de lokale CPU van het eigen werkstation gemiddeld 22.1 msek. Dit is de tijd die het Clientsysteem nodig heeft voor de analyse van de remote-I/O-opdracht, het zenden van een message, het terugontvangen van data/status en voor data moves. De verwerking van de opdracht op de host-CPU kost gemiddeld 17.5 msek. In deze tijd ontrafelt het Serversysteem de message, pleegt een lokale I/O-opdracht, zendt een bericht terug en verplaatst data. De remote-disk heeft bij random acces gemiddeld 68 msek. nodig voor seek, rotational delay en transfer.

verwerkingsduren (msek.)			
CPU lokaal	CPU host	disk serviceduur	responsetijd lokale disk
22.1	17.5	68	310

Bij een bepaald Serversysteem, de Server, worden I/O-opdrachten afkomstig van diverse plaatsen in het net in behandeling genomen. We veronderstellen dat er op die plaatsen I/O-intensieve programmatuur vrijwel continu random acces I/O-opdrachten genereert, de tijd die tussen twee opdrachten door besteed wordt aan de bewerking van de verkregen data zal gering zijn. De opdrachten worden door lokale Clientsystemen aan de Server doorgegeven. Er speelt zich op zo'n plaats

steeds dezelfde cyclus af: er wordt een I/O-opdracht gegenereerd voor de remote-disk, deze opdracht wordt behandeld door de lokale CPU gedurende gemiddeld 22.1 msek., er wordt over het net een message gestuurd naar de host, de CPU in de host is gemiddeld 17.5 msek. bezig met het bericht en de verwerking door de disk duurt gemiddeld 68 msek., message terug over het net en er wordt bijna aansluitend een nieuwe I/O-opdracht gegeven. Het net is snel, zodat de vertraging door de transfer van messages over het net verwaarloosbaar is.

Men vraagt zich af hoeveel I/O-intensieve programma's, die van dezelfde Server gebruik maken, er tegelijk aktief kunnen zijn in het net. De tijd die er verloopt tussen het geven van een remote-I/O-opdracht en het binnenkrijgen van het resultaat noemen we de (remote) I/O-delay. Men vindt dat deze gemiddeld niet mag uitkomen boven de gemiddelde duur van 310 msek. voor een I/O-opdracht op een lokale diskette.

- Het beschreven Client/Server systeem is sterk analoog aan een computersysteem opgebouwd uit een CPU met randapparaten en terminals. Geef aan hoe de overeenkomst precies is. Wat is de denktijd, wat de responsetijd?
- Er maakt maar één I/O-intensief programma gebruik van de server. Geef aan hoe lang de I/O-delay gemiddeld is (107.6).
- De doorstroom aan I/O-opdrachten kan niet hoger worden dan 14.7 opdrachten per sekonde. Hoe is dat te beredeneren?
- Schets het verloop van de gemiddelde I/O-delay als funktie van het aantal aktieve I/O-intensieve programma's in het net. Idem het verloop van de doorstroom.
- Hoeveel I/O-intensieve programma's kunnen er tegelijk van dezelfde Server gebruik maken, gelet op de aangegeven grens van 310 msek. voor de gemiddelde I/O-delay (4)?
- Geef de gemiddelde I/O-delay als er 9 I/O-intensieve programma's gebruik maken van de Server. Welk deel van deze tijd is wachten? Waar wordt gewacht? Hoeveel wachten er gemiddeld?

opgave 6.6: achtergrondcommando's als alternatief

De terminalisten uit opgave 6.3, 'vollopen configuratie', krijgen de mogelijkheid hun opdrachten op de achtergrond te laten draaien. Ze maken ruim gebruik van deze nieuwe faciliteit, omdat ze veronderstellen dat dit voordeel oplevert. Het lukt hen per interaktieve opdracht (een opdracht die niet op de achtergrond draait) gemiddeld één 'achtergrond'-opdracht in te brengen. De gemiddelde denktijd van hun interaktieve opdrachten neemt slechts toe tot 2.5 sekonden, ondanks de extra aandacht die nodig is om het goede verloop van het werk bij te houden.

- Geef de interaktieve doorstroom als in een piektijd alle 220 terminalisten zo handelen. Idem de gemiddelde interaktieve responsetijd (31/58).
- Geef aan hoeveel terminalisten er gemiddeld aan het denken zijn.
- Wat zijn de voordelen, wat de nadelen van de handelwijze?

Beantwoord dezelfde vragen als de gemiddelde denktijd tot 3.2 sekonden zou oplopen en voor 20 ingelogde aktieve terminalisten.

7

Beurten en kansen

7.1. INLEIDING

In de hoofdstukken over de relatie van Little en de responsetijdrelaties gingen we eigenlijk niet verder dan de eenvoudigste vraag wanneer er toestanden worden waargenomen: hoelang treedt een bepaalde toestand achterelkaar op? Er was maar oog voor een en dezelfde toestand, zoals ingelogd of denkend of in response.

Maar het is natuurlijk ook nodig na te gaan hoe de afwisseling van toestanden verloopt, welke toestanden zoal aan bod komen. De eerste vraag daarbij is, hoe vaak —relatief— de verschillende toestanden heersend zijn. De voor de hand liggende volgende vraag is hoe vaak ze worden aangetroffen door wie gaat kijken hoe het ermee gesteld is.

Op die vragen gaan we nu in. Het zal gaan over hoe vaak een systeem in een bepaalde toestand is en hoe vaak het daarin blijkt te zijn. Of, kortweg, over hoe vaak een toestand optreedt en wordt aangetroffen. Het is handig in de formuleringen het alledaagse begrip *beurt* te gebruiken. In plaats van mede te delen dat de toestand Z is, zegt men kortweg neutraal dat Z de beurt† heeft. Het probleem voor dit hoofdstuk is ook: hoe vaak heeft Z de beurt en hoe vaak blijkt Z de beurt te hebben?

Van te voren spreken we precies af vanuit welk standpunt het observeren zal gebeuren: we stellen ons in op een *lukrake* waarnemer, die op volkomen *willekeurige* lukrake —*random*— momenten gaat kijken. Met hem zullen we ons vereenzelvigen.

---

† Beurt heeft niets te maken met voorkeursbehandeling.

We vinden dan de eenvoudige, maar belangrijke betrekking

$$p_Z = b_Z \frac{E(duur_Z)}{E(duur)}$$

Dit is de *basisrelatie* voor de kansen op het optreden en aantreffen van *toestanden*. De kans p_Z op het *aantreffen* van toestand$_Z$ is evenredig met het produkt van de kans b_Z op *optreden* van toestand$_Z$ en de gemiddelde duur $E(duur_Z)$ van optreden van toestand$_Z$. Het produkt van b_Z en $E(duur_Z)$ wordt genormeerd door te delen door de 'overall' gemiddelde toestandsduur $E(duur)$, waardoor de som van de kansen op aantreffen 1 is. De basisrelatie brengt in rekening dat een toestand die langer duurt een evenredig grotere kans heeft om aangetroffen te worden.

Deze fameuze betrekking (de paradox van Kleinrock's hippie of het gevolg van lukraak prikken) leidt in de standaardleerboeken een nogal verscholen leven; soms wordt hij alleen aangehaald bij de mathematische afleiding van de formule voor de gemiddelde wachttijd in het M/G/1 wachttijdensysteem (§15.3). Toch is het een erg praktische stelling, het is steeds nodig de kansen p_Z op 'aantreffen' en b_Z op 'optreden' uit elkaar te houden. De kans op aantreffen is een *'tijdgemiddelde'*; de kans op optreden is een *'aantal-gemiddelde'*.

In de praktijk worden vaak alleen duren waargenomen, b_i is dan de kans op optreden van $duur_i$ en p_i de kans op aantreffen van $duur_i$. De verdeling van de duren wordt aangegeven door de kansen b_i. De betrekking heeft voor duren de eenvoudige vorm

$$p_i = b_i \frac{duur_i}{\sum_j b_j \, duur_j}$$

Dit is de *basisrelatie* voor de kansen op het optreden en aantreffen van *duren*.

7.2. KANSEN OPTREDEN/AANTREFFEN

7.2.1. *kans op optreden*

De aanpak van vragen rond 'hoe vaak is een bepaalde toestand de heersende' is bekend uit de statistiek. Een gemeten beeld van het optreden van de verschillende toestanden van een systeem ontstaat door bij elke wijziging in de toestand de nieuwe toestand te noteren en tellers bij te houden voor alle toestanden van het systeem. Er kan bijvoorbeeld een frequentie-histogram worden getekend. Als er ook sequenties van toestanden geteld worden, kan de correlatie tussen opvolger en voorganger worden uitgezet.

In het eenvoudigste geval loopt het zo: Als er *UIT* toestandswisselingen zijn genoteerd, werden er *UIT* beurten geteld. Wanneer toestand$_Z$ UIT_Z maal optrad is UIT_Z / *UIT* het gemeten relatieve aantal beurten van Z. Het tellen levert dus op hoe vaak Z en de andere toestanden relatief optreden.

Het 'echte' relatieve voorkomen kan nogal eens worden weergeven met een kansverdeling: er is een kans b_Z op het optreden van toestand$_Z$. Het tellen levert informatie over zo'n kansverdeling. Het relatieve aantal beurten UIT_Z / *UIT* is een puntschatting voor de kans b_Z. De uitkomst UIT_Z / *UIT* brengen we in §7.4 in verband met de operationele schatter voor b_Z; men kan daar terecht voor een simpele afleiding van wat we in dit hoofdstuk willen laten zien.

De kansen b_Z op optreden zijn kansen *'op de beurt'*; ze bepalen de verdeling over de toestanden Z en worden gebruikt bij het bepalen van verwachtingswaarde en variantie. Deze kansen op optreden zijn altijd aan het tellen van toestanden —van beurten— gekoppeld, men krijgt ze in handen door ijverig te tellen.

7.2.2. *kans op aantreffen*

We gaan de kansen b_Z nu vergelijken met de kansen p_Z op aantreffen van toestand$_Z$ door een 'lukrake waarnemer'.

Om het denken wat makkelijker te laten verlopen, mag het wel zo zijn dat er altijd een displaytje oplicht, waarop te lezen valt welke toestand wordt bezet. Bij een overgang naar een nieuwe toestand verandert de inhoud van het display. In het display staat behalve de toestand ook steeds hoelang de toestand duurt: naast de naam wordt de tijdsduur tussen het begin en het eind van het huidige optreden aangegeven. Men leest bijvoorbeeld *Z* 10; de toestand is dan *Z* en deze toestand duurt dit keer 10 t.e.

Men gaat op een volledig willekeurig moment eens kijken wat er in het displaytje staat. Als toestand$_Z$ met de kans b_Z de beurt heeft, wordt dan ook in het deel b_Z van de gevallen *Z* aangetroffen? Nee, de kans om het systeem in de toestand *Z* aan te treffen is niet gelijk aan de kans dat *Z* optreedt! De kans p_Z op aantreffen met de beurt hangt ook af van de *duur van de beurt*.

We lopen drie eenvoudige voorbeelden langs. Er zijn steeds drie toestanden *A*, *B* en *Z*. De kansen op optreden en de duren van de toestanden staan in tabel 7.1.

In ons eerste voorbeeld duurt toestand$_A$ altijd 20 t.e. en toestand$_Z$ altijd 10 t.e. Het zullen de enige toestanden zijn die optreden en ze komen gemiddeld even vaak voor, dus de kans op de beurt is voor beide 1/2. Maar er wordt niet even vaak *A* als *Z* gelezen, maar tweemaal zo vaak!

Dat is eenvoudig in te zien door te bedenken dat in de loop van de tijd altijd òf *A* òf *Z* bezet is. Dit verknipt de tijd in stukjes van 20 en 10 t.e. en wel, als de tijdsduur lang is, in vrijwel evenveel stukjes van 10 als van 20. De kans op een stukje van 20 respectievelijk 10 t.e. is immers 1/2. De stukjes van 20 t.e. nemen echter in totaal wel tweemaal zoveel tijd in beslag als de stukjes van 10 t.e. Wordt er zo maar eens gekeken dan komt men gemiddeld tweemaal zo vaak in een stuk van duur 20 als in een stuk van duur 10 terecht. Er wordt dus bij vaak kijken gemiddeld tweemaal zo vaak *A* 20 als *Z* 10 gelezen. De langer durende toestand$_A$

	toestanden			
duur	*A* 20	*B* 30	*Z* 10	
voorbeeld:	**kansen op optreden**			*gem. duur*
1-ste	1/2	0	1/2	15
2-de	1/4	0	3/4	25/2
3-de	1/6	1/3	1/2	55/3

Tabel 7.1. Kansen op optreden: op de beurt.

komt niet vaker voor, maar wel langer en wordt dus ook vaker aangetroffen. De kans op aantreffen van de korte toestand$_Z$ is niet 1/2 maar 1/3. Hier zijn dus $b_A = b_Z = 1/2$, maar $p_A = 2/3$, $p_Z = 1/3$. De kansen op aantreffen zijn samengevat in tabel 7.2 (voor de gemiddelde aangetroffen duur zie §7.5.3).

	toestanden			
duur	*A* 20	*B* 30	*Z* 10	
voorbeeld:	**kansen op aantreffen**			*gem. aangetroffen duur*
1-ste	2/3	0	1/3	50/3
2-de	2/5	0	3/5	14
3-de	2/11	6/11	3/11	250/11

Tabel 7.2. Kansen op aantreffen.

Wat verandert er als *Z* niet even vaak maar driemaal zo vaak als *A* de beurt heeft (tweede voorbeeld)? De tijd wordt dan door de wisseling in de inhoud van het display verknipt in gemiddeld driemaal zoveel stukken van 10 t.e. als van 20 t.e. Gedurende gemiddeld $(3\times10)/50 = 3/5$ deel van de tijd staat er *Z* 10 en gedurende gemiddeld $(1\times20)/50 = 2/5$ deel van de tijd staat er *A* 20. Iemand die lukraak gaat kijken vindt gemiddeld in 2/5 deel van de gevallen toestand$_A$ en in 3/5 deel van de gevallen toestand$_Z$.

De kans b_Z dat *Z* de beurt heeft is 3/4, maar de kans p_Z dat *Z* met de beurt wordt aangetroffen is maar 3/5. De toestand *A* heeft weinig keren de beurt, maar als hij hem heeft houdt hij hem lang. Dat effekt moet in rekening worden gebracht.

Omdat A en Z hier de enige toestanden zijn en Z driemaal vaker de beurt heeft dan A, zal de toestand Z nogal eens door Z als nieuwe toestand worden opgevolgd. Het displaytje knippert dan even, daarna staat er weer hetzelfde als er stond.

In een situatie met 3 toestanden A, B en Z die 20, 30 en 10 t.e. duren en met kansen $b_A = 1/6$, $b_B = 1/3$, $b_Z = 1/2$ de beurt hebben (ons derde voorbeeld), wordt de tijd verknipt in stukjes van 20, 30 en 10 t.e. Als UIT toestanden de beurt gehad hebben, zijn er gemiddeld $UIT/6$ stukjes van duur 20 t.e., $UIT/3$ stukjes van duur 30 t.e. en $UIT/2$ stukjes van duur 10 t.e. Gedurende gemiddeld het deel p_Z van de tijd staat er Z 10, met

$$p_Z = \frac{UIT \times 10/2}{UIT \times (20/6+30/3+10/2)} = 3/11$$

Idem $p_A = 2/11$ en $p_B = 6/11$. Iemand die op een willekeurig moment gaat kijken vindt met een kans van 2/11 toestand_A, met een kans van 6/11 toestand_B en met een kans van 3/11 toestand_Z. Dus niet met kansen van 1/6, 1/3 en 1/2. Hoewel Z met een kans van 1/2 optreedt ($b_Z = 1/2$), wordt hij maar met een kans van 3/11 aangetroffen ($p_Z = 3/11$).

De zaak wordt al duidelijk. Het is kennelijk zo dat de kans op het aantreffen van een toestand zowel evenredig is met de kans dat de toestand optreedt als met de duur van het optreden. De kans wordt vastgelegd —wordt genormeerd— door de eis dat de totale kans gelijk is aan 1.

De analyse van het tweede voorbeeld is dus kortweg zo: De kans op aantreffen van Z is evenredig met de kans op de beurt van 3/4 maal de duur van de beurt van 10 t.e. Die op aantreffen van A is idem evenredig met $1/4 \times 20$. Normeren om de som van de kansen 1 te krijgen, geeft voor de kans p_Z op aantreffen van Z

$$p_Z = (3/4 \times 10)/(1/4 \times 20 + 3/4 \times 10) = 3/5$$

en voor de kans p_A op aantreffen van A

$$p_A = (1/4 \times 20)/(1/4 \times 20 + 3/4 \times 10) = 2/5$$

In het derde voorbeeld is de kans op aantreffen van toestand_A

$$p_A = (1/6 \times 20)/(1/6 \times 20 + 1/3 \times 30 + 1/2 \times 10) = 2/11$$

Het is belangrijk zich te realiseren hoe essentieel het is dat we de vraag naar het aantreffen geïnterpreteerd hebben als: op een lukraak gekozen moment aantreffen. Deze interpretatie is een precisering van wat het gewone spraakgebruik impliciet

bij de vraag veronderstelt: als je zo maar eens gaat kijken, wat vind je dan? Als er niet op lukrake 'willekeurige' momenten wordt geobserveerd zijn de resultaten totaal anders dan bij lukrake waarneming. Wanneer er alleen vlak na een toestandswisseling wordt gekeken vindt men in het derde voorbeeld *A* niet in 2/11, maar in 1/6 deel van de gevallen; in het algemeen vindt men op die momenten toestand$_A$ niet met een kans p_A maar met een kans b_A. Zodra op de een of andere manier de momenten waarop een observatie plaatsvindt gekoppeld, gecorreleerd, zijn met de momenten waarop een toestandswisseling plaatsvindt, wordt er niet gekeken op, ten opzichte van de toestandswisselingen, lukrake momenten. De voorgaande redeneringen en uitkomsten gelden dan niet.

7.2.3. verband optreden/aantreffen voor duren

De bevindingen tot-nu-toe zijn eigenlijk simpel. We zochten 'de kans op aantreffen' en bekeken situaties, waarin de discrete *toestanden$_i$* steeds van duur *duur$_i$* zijn en met een kans b_i optreden. De kans p_i op aantreffen van toestand$_i$ door een 'lukrake waarnemer' blijkt dan te zijn (de som is over alle toestanden *j*)

$$p_i = b_i \frac{duur_i}{\sum_j b_j\, duur_j}$$

Hier staat het verband tussen *optreden* en *aantreffen* voor *duren*. Iedere toestand correspondeert met een duur: p_i is ook de kans op aantreffen van *duur$_i$* en b_i de kans op optreden van *duur$_i$*. Kernpunt is dat de kans op aantreffen wordt bepaald door het genormeerde produkt van de kans op optreden en de duur van het optreden.

Als iedere toestand een bepaalde duur heeft, kan de duur als een identifikatie voor de toestand worden gezien, simpel door te stellen: als de duur gelijk is aan *duur$_i$* is het systeem in toestand$_i$. Er kan natuurlijk altijd aan iedere voorkomende duur een toestand worden gekoppeld. Dergelijke toestanden zijn dan alleen gespecificeerd door de duur van hun beurt. De kansverdeling over de duren *duur$_i$* wordt zo beschreven door de bijbehorende kansen op de beurt b_i! Opeenvolgende duren zullen in het algemeen niet beschouwd kunnen worden als lukrake trekkingen uit de verdeling van de duren, maar dat is ook niet nodig om over een verdeling van de lengten van de duren te kunnen spreken.

Voor discrete duren, met een kans op *duur$_i$* van b_i, is de 'echte' gemiddelde duur gelijk aan

$$E(duur) = \sum_i b_i\, duur_i$$

De gevonden samenhang tussen b_i en p_i kunnen we daarmee nòg compacter opschrijven. De som in de noemer is ook de gemiddelde optredende duur!

De kans op aantreffen van $duur_i$ blijkt dus te zijn

$$p_i = b_i \frac{duur_i}{E(duur)}$$

Dit is de basisrelatie voor de kansen voor het optreden en aantreffen voor duren. Het is voor duren de netste vorm van het verband tussen optreden en aantreffen.

Hoe zien de berekeningen voor het derde voorbeeld er nu uit (tabel 7.3)?

	toestanden			
	A	*B*	*Z*	
	optreden			*gem. duur*
duur	20	30	10	55/3
kans	1/6	1/3	1/2	
	aantreffen			
kans	2/11	6/11	3/11	

Tabel 7.3. Verdeling duren derde voorbeeld.

Er zijn duren van 10, 20 en 30 t.e., de kansen dat deze duren de beurt hebben zijn $b_Z = 1/2$, $b_A = 1/6$ en $b_B = 1/3$. De kansverdeling over de duren is†: $b_{10} = Pr(duur = 10) = 1/2$, $b_{20} = Pr(duur = 20) = 1/6$ en $b_{30} = Pr(duur = 30) = 1/3$. De gemiddelde duur is

$$E(duur) = b_{10}\, duur_{10} + b_{20}\, duur_{20} + b_{30}\, duur_{30}$$

$$= 1/2 \times 10 + 1/6 \times 20 + 1/3 \times 30 \quad = 18\frac{1}{3}$$

De uitkomst voor p_A is te schrijven als:

$$p_{20} = (b_{20}\, duur_{20}) / (b_{10}\, duur_{10} + b_{20}\, duur_{20} + b_{30}\, duur_{30})$$

en dat is

$$p_{20} = b_{20}\, duur_{20} / E(duur)$$

---

† *Pr* is 'kans op' (Probability)

7.2.4. voorbeeld: data, code, kernel

Een processor verwerkt instrukties van het type 'data', van het type 'code' en van het type 'kernel'. Hij doet over een instruktie van type data precies 1/16 msek., over een instruktie van type code precies 1/8 msek. en over een instruktie van type kernel precies 1/32 msek. De aanstroom aan instrukties, en dat is omdat alle instrukties in eindige tijd verwerkt worden ($IN = UIT$) ook de doorstroom aan instrukties, bestaat gemiddeld uit 20% data-, 40% code- en 40% kernelinstrukties (verhouding 1:2:2). Zie ook tabel 7.4 (voor C^2 zie §7.7.2).

processor	type instruktie				
	data	**code**	**kernel**		
duur	1/16	1/8	1/32		
	kans op duur			*gem. duur*	C^2
	1/5	2/5	2/5	3/40	23/72
	kans op aantreffen				
executeert	1/6	2/3	1/6		
kans idle 1/4	1/8	1/2	1/8		

Tabel 7.4. Data, code, kernel.

De processor krijgt te maken met verwerkingsduren van 1/16, 1/8 en 1/32 msek. Deze treden op met kansen 1/5, 2/5 en 2/5 of b_{data} = 1/5, b_{code} = 2/5 en b_{kernel} = 2/5 (dit is dus hier de relevante vertaling van het gegeven 20% data, 40% code en 40% kernel, vergelijk bij twijfel de discussie rond de splitsing naar groep bij de schatter voor de doorstroom in het hoofdstuk RESPONSETIJDRELATIES, §5.6.1). De kansverdeling van de verwerkingsduren is daarmee bekend; de kansen b op de duren beschrijven deze verdeling. Het is een discrete verdeling over de mogelijke waarden 1/32, 1/16 en 1/8 met bijbehorende kansen 2/5, 1/5 en 2/5. De gemiddelde verwerkingsduur is volgens het adagium 'som van waarde maal kans-op-waarde'

$$E(verwerkingsduur) = 1/5 \times 1/16 + 2/5 \times 1/8 + 2/5 \times 1/32 = 3/40 \text{ msek.}$$

De processor is op een bepaald moment bezig. Hoe groot is de kans dat hij data verwerkt? Die kans is niet: in 20 van de 100 gevallen, naar de kansen op de beurt. Nee, het is de kans op het aantreffen van 'bezig met data'. We interpreteren 'op een bepaald moment' als op een willekeurig moment, zoals het gewone spraakgebruik wil. Het bepaalde moment heeft niets bijzonders immers. We kunnen

daarom het gevonden verband gebruiken. De kans dat de processor met data bezig is, bedraagt

$$p_{data} = 1/5 \times \frac{1/16}{1/5 \times 1/16 + 2/5 \times 1/8 + 2/5 \times 1/32}$$

of

$$p_{data} = 1/5 \times \frac{1/16}{E(verwerkingsduur)} = 1/5 \times \frac{1/16}{3/40} = 1/6$$

Net zo is $p_{code} = 2/3$, $p_{kernel} = 1/6$. Merk op dat de kansen op aantreffen hier volgen uit de verdeling van de verwerkingsduren; dat zijn in dit geval de toestandsduren, we volgen dus de herleiding uit de vorige paragraaf. In zo'n verdeling zit zowel het gegeven 'kans op de toestand/beurt' als de duur van die toestand/beurt; in een verdeling staan de 'kansen op optreden'.

Hoe groot is de kans *zonder meer* dat de processor data verwerkt? Die is alleen aan te geven als bekend is hoe sterk de processor bezet is. Laten we aannemen dat de processor 3/4 van de tijd bezig met het verwerken van instrukties (bezettingsgraad 0.75). Dan is er een kans van 1/4 om de processor zonder werk (idle) aan te treffen en 3/4 hem bezig te vinden, als er zo maar eens gekeken wordt. Dus $p_{idle} = 1/4$. De kans hem met dataverwerking bezig te vinden is $3/4 \times 1/6 = 1/8$. In totaal vinden we nu als kansen op aantreffen $p_{idle} = 1/4$, $p_{data} = 1/8$, $p_{code} = 1/2$, $p_{kernel} = 1/8$. Zojuist was de betekenis van p_{data} de kans om 'data' aan te treffen onder de voorwaarde dat de processor bezig is. Nu is p_{data} de kans de processor met 'data' bezig te vinden, zonder die voorwaarde. In dit geval is er naast de toestanden 'code', 'data' en 'kernel' ook nog de toestand 'idle', waarin de processor niet bezig is met de verwerking van de instrukties. Zo'n veralgemening kan pijnloos worden meegenomen.

Als we ons aan de letter van de eerdere redenering rond aantreffen/optreden houden, mogen de verwerkingsduren niet onderbroken worden, de afwikkelingsvolgorde zou bijvoorbeeld FCFS kunnen zijn. Maar die redenering kan natuurlijk direkt uitgebreid worden naar ook onderbroken toestandsduren, zoals bij een 'preemptive' verwerkingsvolgorde als ROUND ROBIN. De betrekkingen voor de kansen op aantreffen gelden ook dan, ze zijn heel algemeen.

7.3. VERDELING DUREN EN KANSEN OP AANTREFFEN

7.3.1. *per toestand spreiding in duur*

Meestal zullen toestanden niet volledig gekarakteriseerd worden door hun toestandsduur. Gewoonlijk zal er bij een bepaalde toestand een verdeling optreden over de duur van de toestand. Ook dan is de kansverdeling over de duren aan te geven door de verdelingen van de duren van de toestanden te combineren met de

kansen op optreden van deze toestanden. Deze 'overall' verdeling over de duren volgt simpelweg na vermenigvuldigen en optellen van kansen op dezelfde duur. Laten we om dit te illustreren nog een laatste voorbeeld langslopen (tabel 7.5).

	toestanden					
gem. duur	*A* *20*		*B* *30*		*Z* *10*	
	kansen op de beurt					*gem. duur*
	1/6		1/3		1/2	55/3
	optreden					
vaste duur	15	30	20	80	10	
optreden	2/3	1/3	5/6	1/6	1	
kans	1/9	1/18	5/18	1/18	1/2	55/3
	aantreffen					
kans	1/11	1/11	10/33	8/33	3/11	
	2/11		6/11			

Tabel 7.5. Spreiding in duren (uitbreiding derde voorbeeld).

We passen ons derde voorbeeld uit tabel 7.1 wat aan. De toestand A zal nu niet een duur van 20 hebben, maar in een derde deel van de gevallen een duur van 30 en in twee-derde deel van de gevallen een duur van 15. De gemiddelde duur van A is dan nog steeds 20 = $1/3 \times 30 + 2/3 \times 15$. Maar er zit nu spreiding in de duur van toestand$_A$. Idem zal toestand$_B$ met een kans van 5/6 duur 20 en met een kans van 1/6 duur 80 hebben, met als gemiddelde duur nog steeds 30. De duur van toestand$_Z$ blijft strikt 10 t.e. De kans op toestand$_A$ blijft 1/6, op toestand$_B$ 1/3 en op toestand$_Z$ 1/2; nog steeds zijn de kansen op de beurt: $b_A = 1/6$, $b_B = 1/3$ en $b_Z = 1/2$.

De 'overall' gemiddelde duur blijft 55/3:

$$E(duur) = 1/6 \times 20 + 1/3 \times 30 + 10/2 = 18\frac{1}{3}$$

De kansverdeling over de duren is als volgt. Er zijn duren van 10, 15, 20, 30 en 80 t.e. De kans op duur 15 is $(b_A \times 2/3) = 1/9$, de kans op duur 20 is $(b_B \times 5/6) = 5/18$, enzovoort. In totaal is $Pr(duur=10) = 1/2$, $Pr(duur=15) = 1/9$, $Pr(duur=20) = 5/18$, $Pr(duur=30) = 1/18$ en $Pr(duur=80) = 1/18$. Met ook

$$E(duur) = 1/2 \times 10 + 1/9 \times 15 + 5/18 \times 20 + 1/18 \times 30 + 1/18 \times 80 = 18\frac{1}{3}$$

De duren 10, 15, 20, 30 en 80 treden op met kansen $b_{10} = 1/2$, $b_{15} = 1/9$, $b_{20} = 5/18$, $b_{30} = 1/18$ en $b_{80} = 1/18$. Een verdeling van duren wordt uiteindelijk altijd beschreven met b's.

De kans om duur 15 aan te treffen is volgens het verband tussen optreden/aantreffen voor duren

$$p_{15} = (1/9 \times 15)/(55/3) = 1/11$$

7.3.2. verband optreden/aantreffen voor toestanden

In het voorbeeld uit tabel 7.5 heeft toestand$_A$ niet een vaste duur, de duur is 15 met kans 2/3 en 30 met kans 1/3. Wat is nu de kans A aan te treffen? Met de eerdere redenering komen we op

$$p_A = 1/6 \times \frac{1/3 \times 30 + 2/3 \times 15}{(30/3+30/3)/6+(100/6+80/6)/3+10/2}$$

of opnieuw $p_A = 2/11$. De termen in teller en noemer zijn bekenden. De teller is de gemiddelde duur van de toestand; de noemer is de gemiddelde duur over alle toestanden. Er staat eigenlijk:

$$p_A = b_A \frac{E(duur_A)}{b_A\, E(duur_A) + b_B\, E(duur_B) + b_Z\, E(duur_Z)}$$

of

$$p_A = b_A \frac{E(duur_A)}{E(duur)}$$

Het is gewoon een generalisatie van het eerder in OPTREDEN/AANTREFFEN VOOR DUREN (§7.2.3) gegeven verband.

We kennen nu de algemene relatie tussen de kansen op optreden en aantreffen van toestanden. Als er verschillende toestanden zijn, met elk een verschillende gemiddelde duur, geldt voor de kansen b_i op het optreden van toestand$_i$ en p_i op het aantreffen van toestand$_i$

$$p_i = b_i \frac{E(duur_i)}{E(duur)}$$

met $E(duur_i)$ als de gemiddelde duur van toestand$_i$ en $E(duur)$ als de overall gemiddelde duur van een toestand. Dit is de meest algemene vorm van het verband tussen de kans op optreden en de kans op aantreffen; het is de *basisrelatie* voor de kansen op optreden en aantreffen van *toestanden*.

7.3.3. continue verdelingen (mathematisch intermezzo)

Heel vaak zijn de kansverdelingen voor de duren continu: alle mogelijke duren —tussen de minimale duur ($\geqslant 0$) en de maximale duur— treden op. Ook voor een continue kansverdeling over de duren schrijven we de bevindingen nu uit. De kans op het optreden van een duur met een lengte 'tussen x en $x+dx$ in' zij $f(x)dx$; $f(x)$ is de kansdichtheid voor de continue verdeling over de toestandsduren (en dat is dus ook een continue verdeling over toestanden). De kans om zo'n duur op een lukraak moment aan te treffen is evenredig met de lengte van de duur x en met de kans $f(x)dx$ op de toestand —hier dus op de duur tussen x en $x+dx$. We noemen die kans $g(x)dx$. De evenredigheidsconstante wordt vastgelegd door de eis dat de kans om op een moment iets aan te treffen 1 is. Dus moet de 'kans(dichtheid) op het aantreffen van duur x' gelijk zijn aan

$$g(x) = x f(x) / \int dx\, x f(x)$$

In de noemer staat de integraal van $x f(x)$ over heel het gebied van mogelijke duren. De totale integraal over x van linker en rechterlid is 1.

De integraal in de noemer is de rechtstreekse generalisatie van de som in de noemer uit de voorgaande discrete voorbeelden. Deze noemer is ook weer de verwachtingswaarde $E(duur)$ van de duur van een toestand, zodat

$$g(x) = x f(x) / E(duur)$$

Tussen de kans op het aantreffen van een duur en de kans op een duur bestaat zowel in het continue geval als in het discrete geval dezelfde relatie.

Dit geldt ook als we per toestand een verdeling voor de duren onderscheiden. In de basisrelatie voor de kansen op optreden en aantreffen van toestanden

$$p_i = b_i \frac{E(duur_i)}{E(duur)}$$

is als de duren continu zijn

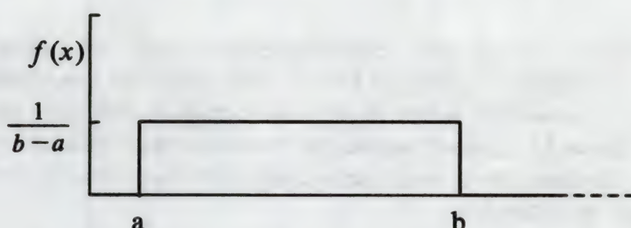
$$E(duur) = \int dx\ x f(x)$$

$$E(duur_i) = \int dx\ x f_i(x)$$

Hierin beschrijft $f(x)$ de 'overall' kansverdeling, terwijl $f_i(x)$ de kansverdeling beschrijft van de tijdsduren waarin toestand$_i$ de heersende toestand is.

7.3.4. voorbeeld: berichten en ack's

Over een kanaal worden berichten verstuurd van alle mogelijke lengten tussen 300 bits en 3000 bits. De snelheid van het kanaal is 30 Kbits/sek. De transfertijden liggen dus tussen 10 en 100 msek. Veronderstel dat de verdeling over de transfertijden uniform is: alle duren tussen 10 en 100 msek. komen gemiddeld even vaak voor.



Figuur 7.1. Kansdichtheidsfunktie voor uniforme verdeling op [a,b].

Kenmerkend voor de uniforme verdeling is dat de kansdichtheidsfunktie constant is en wel hier

$$f(x) = 1/(100-10)$$

omdat natuurlijk de (totale) kans op een transfertijd 1 moet zijn.

$$\int dx\, f(x) = \int_{10}^{100} dx\, 1/(100-10) = 1$$

De gemiddelde transfertijd is

$$E(transfertijd) = \int_{10}^{100} dx\, x\, 1/(100-10) = 55$$

uiteraard. De variantie in de transfertijd is (msek.^{2})

$$\int_{10}^{100} dx\, (x-55)^2/(100-10) = (100-10)^2/12$$

Dit zijn de gewone uitdrukkingen voor verwachtingswaarde en variantie van een uniforme verdeling op een interval van a naar b; $a = 10$, $b = 100$.

	berichten en ack's		
type	gem. duur	kans op optreden	kans op aantreffen
bericht	55	10/11	100/101
ack	5.5	1/11	1/101

Tabel 7.6. Berichten en acknowledgements.

Naast deze berichten worden ook acknowledgements over het kanaal verstuurd. Deze ack's zijn gemiddeld 10 maal zo kort als berichten: de gemiddelde transfertijd is 5.5 msek. Gemiddeld wordt er op tien berichten één ack verstuurd. Er wordt op zeker (lukraak) moment met een hardware-monitor geconstateerd dat het kanaal bezet is. De kans dat er een acknowledgement loopt is volgens de uitdrukking voor de kans op aantreffen:

$$p_{ack} = \frac{1/11 \times 5.5}{1/11 \times 5.5 + 10/11 \times 55} = 1/101$$

Dit antwoord ligt voor de hand. Als er per ack 10 berichten lopen, die gemiddeld 10 maal zo lang zijn als een ack, is de kans een ack te treffen $1 \times 1/(1 \times 1 + 10 \times 10) = 1/101$.

7.3.5. voorbeeld: gebundelde Remote-write's

Een station uit een computernetwerk krijgt blokken data aangeleverd, die elders opgeslagen moeten worden. De blokken worden doorgegeven via een op het station draaiend serverproces. Elk binnenkomend blok meldt zich bij de server door een remote-write request. De afhandeling is als volgt: Wanneer een request binnenkomt op een moment waarop de server vrij is, wordt er een Remote-write-1 operatie geïnitieerd: de netwerkdriver gaat aan het werk. Deze voert een programma uit waarin de data over het net getransporteerd worden. De server blijft bezet tot de netwerkdriver klaar is en het blok is weggeschreven.

Requests die binnenkomen op momenten waarop de server bezig is, moeten wachten tot de server klaar is met de vorige Remote-write; ze worden in een lijst van wachtenden gehangen. De gezamenlijke requests van wachtende data worden gebundeld zodra de server vrij komt en er wordt één enkele Remote-write operatie voor geïnitieerd (bij i wachtende requests een Remote-write-i operatie). De server blijft bezet tot de i blokken via de netwerkdriver zijn weggeschreven. De remote-write requests genereren dus Remote-write operaties en het hangt van de omstandigheden af, welke. We onderscheiden de mogelijke Remote-write's als Remote-write-i operaties.

Er blijken gemiddeld 83 Remote-write-1 operaties te worden uitgevoerd tegen 13 Remote-write-2 operaties en 4 Remote-write-3 operaties. Remote-write-i operaties

server	**Remote-write**			
request gem. duur	-1- 30	-2- 40	-3- 50	
	kans op **optreden**			*gem. duur*
	0.83	0.13	0.04	32.1
idle	kans op **aantreffen**			
0.20	0.62	0.13	0.05	

Tabel 7.7. Remote-write requests.

met $i \geqslant 4$ treden niet op. Remote-write requests ontstaan op lukrake momenten. De server is 20% van de tijd werkeloos, dat wil zeggen niet bezet met Remote-write operaties. De verwerking van een Remote-write, afkomstig van 1 request, duurt gemiddeld 30 msek., die afkomstig van 2 requests gemiddeld 40 msek., van 3 requests 50 msek., enzovoort. Een Remote-write-i operatie wordt dus verwerkt in gemiddeld $30+10\times(i-1)$ msek., een lineair verband gemakshalve. De winst van de bundeling van requests zit in het verminderen van de initiële overhead bij de server (en in het gebruiken van maar één header bij het oversturen).

De informatie over de verdeling van de duren van Remote-write-operaties is dus als volgt. Er zijn duren in toestand Remote-write-1 van gemiddeld 30 ($E(duur_1) = 30$), in toestand Remote-write-2 van gemiddeld 40 ($E(duur_2) = 40$) en in toestand Remote-write-3 van gemiddeld 50 msek. ($E(duur_3) = 50$). Deze toestanden treden op met kansen van $b_1 = 0.83$, $b_2 = 0.13$ en $b_3 = 0.04$. De gemiddelde duur van een Remote-write operatie is

$$E(duur) = 0.83\times30+0.13\times40+0.04\times50 = 32.1 \text{ msek.}$$

De kans dat een Remote-write request binnenkomt op een moment dat de server bezig is, is 0.80; de kans dat hij binnenkomt op een moment dat er een Remote-write-1 operatie loopt is

$$p_1 = 0.80\times(0.83\times30)/32.1 = 0.62$$

Idem $p_2 = 0.13$ voor Remote-request-2 en $p_3 = 0.05$ voor Remote-request-3. De kansen op aantreffen zijn $(p_{idle}, p_1, p_2, p_3) = (0.20, 0.62, 0.13, 0.05)$.

7.4. OPERATIONELE SCHATTERS VOOR KANSEN

Hoewel we nu uitvoerig zijn ingegaan op de samenhang tussen de beide typen kansen —en de verrassing voorbij is—, willen we toch kijken wat een operationele analyse oplevert. De operationele aanpak brengt ons rechtstreeks bij de gevonden verbanden!

Er zal gedurende een tijdsduur van duur *MEETDUUR* gemeten worden aan een bepaald systeem. Als deze duur verstreken is wordt er geteld wat er is gebeurd. Er blijkt *UIT* maal een overgang te zijn gemaakt naar een andere toestand, waarvan UIT_Z maal vanuit de toestand$_Z$. Gedurende een (niet noodzakelijk aaneengesloten) tijdsduur $WERK_Z$ blijkt *Z* de beurt te hebben (er stond in die tijd *Z* in het display).

De operationele schatter voor de kans dat *Z* de beurt heeft is

$$'b_Z = UIT_Z \, / \, UIT$$

Het gemeten relatieve aantal beurten voor *Z* wordt dus gezien als de gemeten waarde van de schatter voor de kans op optreden. Deze operationele schatter is analoog aan de operationele schatter voor de 'echte' gemiddelde duur in het hoofdstuk INLEIDING OPERATIONELE ANALYSE (§3.4); het verband blijkt door een kans te zien als de verwachtingswaarde van een stochast van het 'indicator' type.

De operationele schatter voor de kans dat *Z* wordt aangetroffen door iemand die op een lukraak moment gaat kijken, wordt gedefinieerd als

$$'p_Z = WERK_Z \, / \, MEETDUUR$$

Deze schatter leerden we ook kennen in §3.4.

De operationele schatter voor de gemiddelde duur van een beurt van toestand$_Z$ is volgens diezelfde inleiding

$$'duur_Z = WERK_Z \, / \, UIT_Z$$

Tenslotte is de operationele schatter voor de overall gemiddelde duur van een beurt (steeds heeft tijdens de meetduur een toestand de beurt)

$$'duur = MEETDUUR \, / \, UIT$$

Als de duren van toestand$_Z$ een spreiding hebben is $'duur_Z$ een schatter voor de verwachtingswaarde $E(duur_Z)$ van $duur_Z$.

Als elke beurt van toestand$_Z$ steeds d_Z t.e. duurt, zoals in de voorbeelden 1, 2 en 3 uit tabel 7.1, zal ook de gemeten waarde voor $'duur_Z$ vrijwel precies d_Z zijn. De waarde is niet exact d_Z, doordat aan het begin en het eind van de meetperiode in

$WERK_Z$ ook beurten kunnen worden meegenomen, die maar gedeeltelijk binnen de meetduur vallen.

De schatter *'duur* schat het 'overall' gemiddelde van de gemiddelde duren van een beurt, dus gemiddeld over alle toestanden Z en binnen die toestanden over alle mogelijke duren. Merk op dat er voor het schatten van de gemiddelde duur geen toestandsduren hoeven te worden bijgehouden (§3.4).

Tussen deze operationele schatters bestaat altijd een verband. We schrijven de uitdrukking voor $'p_Z$ als

$$'p_Z = WERK_Z / MEETDUUR$$

$$= (UIT_Z / UIT)(WERK_Z / UIT_Z) / (MEETDUUR / UIT)$$

en vinden

$$'p_Z = 'b_Z \frac{'duur_Z}{'duur}$$

Dit is vooreerst niet anders dan een relatie die aangeeft dat de 4 operationele schatters $'p_Z$, $'b_Z$, $'duur_Z$ en $'duur$ niet onafhankelijk zijn. Er moet tussen meetresultaten deze samenhang bestaan; dit is de *beperkte* vorm van het verband (vergelijk §4.4 en §5.4).

Wanneer de metingen te zien zijn als uitkomsten van stochastische variabelen, waarvan de verwachtingswaarden geschat worden door de operationele schatters, kan de verkregen relatie mogelijk worden uitgebreid tot een relatie tussen verwachtingswaarden: de *uitgebreide* vorm van de relatie. Deze vorm is

$$p_Z = b_Z \frac{E(duur_Z)}{E(duur)}$$

waarin p_Z en b_Z echte kansen zijn en $E(duur_Z)$ en $E(duur)$ echte gemiddelden, dus verwachtingswaarden. De relatie in deze uitgebreide betekenis hebben we in §7.3.2 vanuit een andere invalshoek 'ontdekt'.

De zo op vele manieren gevonden samenhang tussen de kans op optreden en de kans op aantreffen is duidelijk een echte basisrelatie!

Ook nu weer bewijst de operationele aanpak zijn nut voor het schouwen van de wezenlijke samenhang. Met de operationele aanpak is direkt het resultaat gevonden voor de ruimste beschrijving: voor het verband tussen de kansen op optreden en aantreffen voor toestanden. Bovendien laat het zien hoe er over verdere parameters ('de omstandigheden') gemiddeld wordt.

De ongecompliceerde handelwijze van de operationele aanpak is really charming. De eenvoudige behandeling van randeffekten blijkt effektief; gedeelten worden gewoon òf volledig òf helemaal niet meegeteld. Toch geldt ook hier dat statistische en stochastische beschouwingen nodig zijn om de omstandigheden te vinden waaronder een operationeel gevonden relatie in de uitgebreide vorm geldig en zinvol is.

7.5. TOEPASSINGEN

7.5.1. *algemene vorm E(duur)*

Uit de verdeling van de duren bij de verschillende toestanden volgt een 'overall' verdeling van de duren met een 'overall' gemiddelde. Er moet dus verschil worden gemaakt tussen het 'overall' gemiddelde $E(duur)$ en het individuele gemiddelde $E(duur_Z)$. De overall gemiddelde duur $E(duur)$ zal een gewogen gemiddelde zijn van de individuele gemiddelden $E(duur_Z)$. Er geldt natuurlijk (we gebruikten het al; de som is over de toestanden Z)

$$E(duur) = \sum_Z b_Z \, E(duur_Z)$$

Laten we dit —ter oefening— met de operationele aanpak nog eens controleren. Er wordt gebruikt dat $MEETDUUR = \sum_Z WERK_Z$.

$$'duur = MEETDUUR\,/\,UIT = \sum_Z (WERK_Z\,/\,UIT_Z)(UIT_Z\,/\,UIT)$$

$$= \sum_Z {'b_Z}\ {'duur_Z}$$

7.5.2. *voorbeeld: bezet en onbezet*

Een of ander apparaat is ofwel bezet ofwel vrij. De perioden van bezet zijn en vrij zijn wisselen elkaar af. Heeft vrij de beurt, dan krijgt daarna bezet de beurt, waarna weer de beurt aan vrij is. De toestanden 'bezet' en 'vrij' zijn elkaar uitsluitende toestanden. De kans b_{vrij} dat vrij de beurt heeft is 1/2 en ook de kans b_{bezet} dat bezet de beurt heeft is 1/2.

Laten de perioden van ononderbroken bezet zijn een verdeling hebben met verwachtingswaarde $E(bezet)$ en de perioden van vrij zijn een verdeling met verwachtingswaarde $E(vrij)$. De verwachtingswaarde van de duur van een beurt is dan $(E(bezet) + E(vrij))/2$. We noemen de kans om het apparaat op een lukraak moment in de toestand 'bezet' aan te treffen U, U is de *bezettingsgraad* (of

utilisatie) van het apparaat. Volgens de basisrelatie voor de kansen op optreden/aantreffen van toestanden (§7.3.2) geldt (de halven vallen weg)

$$U = E(bezet)/(E(vrij)+E(bezet))$$

Wanneer er twee alternatieven zijn, geldt deze voor de hand liggende relatie. Als een machine gemiddeld 100 t.e. achterelkaar in bedrijf is en gemiddeld 10 t.e. achterelkaar 'plat', is de kans dat hij in bedrijf is $U = 10/11$.

7.5.3. *aangetroffen duur*

De basisrelatie geeft de kans om duren aan te treffen. Hoe lang is de aangetroffen duur gemiddeld? De verwachtingswaarde van de aangetroffen duur is bij discrete duren

$$E(aangetroffen\ duur) = \sum_i p_i\, duur_i = \frac{\sum_i b_i\, duur_i^{\,2}}{\sum_i b_i\, duur_i}$$

en bij continue

$$E(aangetroffen\ duur) = \int x^2 f(x) / \int x f(x)$$

In beide gevallen is dit kortweg

$$E(aangetroffen\ duur) = E(duur^2)/E(duur)$$

De gemiddelde aangetroffen duur is heel anders dan de gemiddelde duur! In het volgende hoofdstuk komen we in wezen dezelfde herleiding opnieuw tegen bij de gemiddelde restduur, we gaan daar wat meer in op de gedachtengang.
In de tabellen 7.2 en 8.3 staan naast de gemiddelde duren ook de gemiddelde aangetroffen duren.

7.5.4. *voorbeeld: pollen*

Een processor pol(l)t voortdurend een aantal stations of ze misschien data hebben. Als een station meldt dat het data heeft, haalt de processor aansluitend de data over. Daarop wordt aan het volgende station gevraagd of er data zijn, enzovoort. De toestanden van de processor zijn dus 'bezig met station zonder data' en 'bezig met station met data', of kortweg *poll* en *transfer*.

In de toestand 'poll' polt de processor het station en krijgt de boodschap terug dat er geen data zijn. In de toestand 'transfer' polt de processor het station, merkt

	pollen			
gem. duur ($/ duur_{poll}$)	transfer 4	transfer-item 4/3	poll 1	
	optreden			*gem. duur*
kans	6/13	18/25	7/13 7/25	31/13
	aantreffen			
kans	24/31	24/31	7/31	

Tabel 7.8. Pollen.

dat er data zijn en haalt de data over. Toestand 'transfer' treedt gemiddeld op in 6 van de 13, toestand 'poll' in 7 van de 13 gevallen; $b_{transfer} = 6/13$, $b_{poll} = 7/13$.

De data bij een station zijn daar binnengekomen sinds de laatste maal dat het station werd gevraagd. Het zijn een aantal items, afkomstig van verschillende bronnen. Deze worden in de toestand 'transfer' één voor één overgehaald (exhaustive transfer). Toestand 'transfer' bestaat dan ook uit een sequentie van een of meer subtoestanden 'transferitem'; als er drie items zijn, bestaat 'transfer' uit driemaal 'transferitem' achterelkaar (de tijd verbruikt om te vragen of er data zijn wordt niet apart gehouden).

We zouden dus ook de toestanden 'transferitem' en 'poll', in plaats van 'transfer' en 'poll' kunnen onderscheiden. Dit heeft pas zin als we weten uit hoeveel subtoestanden toestand 'transfer' gemiddeld bestaat.

Veronderstel dat 'transfer' gemiddeld uit 3 keer achterelkaar subtoestand 'transferitem' is opgebouwd. Er worden dus gemiddeld 3 items overgestuurd. Tegen 7 beurten 'poll' staan dan gemiddeld 6×3 beurten 'transferitem'. De kans dat 'poll' de beurt heeft daalt van 7/13 tot $7/(7+18) = 7/25$. De kans dat 'transferitem' de beurt heeft is 18/25.

We kunnen dit ook formuleren als: de kans dat een subtoestand van 'transfer' de beurt heeft is 18/25. Dan lijkt het alsof 'transfer' nu veel vaker de beurt heeft, maar dat is maar schijn. Er zal vrij vaak een wisseling van 'transferitem' naar 'transferitem' optreden.

Hoe groot is de kans om toestand 'transferitem' aan te treffen, vergeleken met de kans om toestand 'transfer' aan te treffen?

Als toestand 'transfer' gemiddeld viermaal zo lang is als toestand 'poll', is de kans om 'transfer' aan te treffen

$$p_{transfer} = (6\times4)/(6\times4+7\times1) = 24/31$$

Toestand 'transferitem' zal gemiddeld 4/3 maal zo lang zijn als toestand 'poll' en

$$p_{transferitem} = (18\times 4/3)/(18\times 4/3+7\times 1) = 24/31$$

Uiteraard zijn $p_{transfer}$ en $p_{transferitem}$ gelijk.

7.6. METINGEN MET SOFTWARE-MONITORS

Gegevens over het funktioneren van een computersysteem kunnen worden verkregen door een software-monitor te gebruiken (voor nut en gebruik van een software-monitor zie het hoofdstuk ACHTERGRONDEN, §2.3.2). Bij het ontwikkelen van een software-monitor wordt gebruik gemaakt van twee grondgedachten. We geven van beide een simpel voorbeeld.

- In het operatingsysteem worden op diverse plaatsen schrijfopdrachten aangebracht die relevante informatie wegschrijven, wanneer het betrokken stuk code wordt doorlopen. Zo kan bijvoorbeeld het aantal malen bepaald worden dat de processor de instruktie TRAP uitvoert. Dit aantal levert een gemeten waarde voor b_{trap}.
- Om de zoveel tijdseenheden (tikken) wordt via een hardware-interrupt van de (systeem)clock het proces 'demon' gestart, dat de status van de randprocessor wegschrijft. Het is demon die telkens komt kijken hoe het staat met de status van de randprocessor. Dit levert een meting op van p_{status}, tenminste als demon op willekeurige, lukrake, momenten kijkt.

De eerste methode is de *interceptietechniek*. Iedere meetwaarde wordt meestal voorzien van een tijdindicatie. De gemeten waarde kan stante pede worden weggeschreven via een interne interrupt, maar het geeft minder meetoverhead als gegevens ter plaatse enige tijd kunnen worden opgepot. De informatie uit vele sequentiële meetpunten wordt dan af en toe weggeschreven naar een goed bereikbare plaats, uiteindelijk naar een file waar de totale voorraad gegevens bewerkt kan worden met statistische technieken en waar datareduktie kan plaatsvinden.

De tweede methode is de *bemonsteringstechniek*. Hierbij wordt met constante regelmaat een meting gedaan aan de toestand van het systeem. Ook hier komen de gegevens uiteindelijk op een file terecht. Een bekend probleem bij deze methode is het afgeschermd zijn van delen van het operatingsysteem, ze laten zich niet interrumperen door een clock. De clockinterrupt blijft dan enige tijd 'hangen', daarvoor zal moeten worden gecompenseerd.

IBM levert op haar mainframes als software-monitor zowel bijvoorbeeld RMF (Resource Measurement Facility), die met bemonstering werkt, als bijvoorbeeld GTF (Generalised Trace Facility), die met interceptie werkt. Een zelf-geschreven software-monitor volgens de bemonsteringmethode kan door de gebruiker worden geïmplementeerd via een voorziening die bij de diverse fabrikanten 'uiteraard'

diverse namen heeft; bij de PDP-11'en heette het device 'programmeerbare klok', bij IBM interval timer.

Met de interceptietechniek vinden we de kansen op het optreden van toestanden en met de bemonsteringstechniek de kansen op het aantreffen van de toestanden. Bij de presentatie van meetresultaten, die met de bemonsteringstechniek verkregen zijn, wordt vaak verbazing geuit als blijkt dat heel vaak optredende toestanden nauwelijks in het waarnemingsmateriaal voorkomen. Men verwacht met een grote kans (b_Z) toestand$_Z$ te vinden en treft een kleine kans (p_Z) toestand$_Z$ aan: foute meting? Nee dus, maar Z duurt relatief kennelijk erg kort.

Maar mag men voor het duiden van de met de bemonsteringstechniek verkregen waarnemingen inderdaad aannemen dat zo'n 'demon' op lukrake momenten kijkt? Eigenlijk kijkt demon op momenten die vaste tussenpozen hebben. In de praktijk zijn deze momenten ten opzichte van wat er gebeurt, toch wel als willekeurig te beschouwen. Dit is bij de bepaling van p_{status} het geval als de momenten waarop demon wordt doorgestart door de systeemclock niet correleren met de momenten waarop de status van de randprocessor zich wijzigt. De mate van correlatie kan uit het waarnemingsmateriaal worden bepaald met bekende statistische technieken.

Als er een duidelijke correlatie aanwezig is, zijn de met de monitor verkregen waarden voor p_{status} onbetrouwbaar. Demon kijkt in dat geval zeker niet op willekeurige momenten. Het is zaak om altijd bedacht te zijn op het optreden van deze onderlinge afhankelijkheid bij het ontwikkelen en gebruiken van een monitor, die werkt volgens bemonstering. Statistische analyse van het waarnemingsmateriaal is geen luxe.

We zouden de tweede methode ook kunnen beschrijven als een voorbeeld van een *sampling* techniek. Er wordt een lukraak getrokken steekproef van waarden bepaald. De 'sampling' techniek is erg gebruikelijk bij technische metingen. In plaats van continu te meten wordt er af en toe gemeten, 'gesampled'. En heel vaak betekent dit 'af en toe': periodiek. Ook hier zijn dan weer misverstanden. Als de frequentie waarmee gesampled wordt veel hoger ligt dan de mate waarin er toestandsveranderingen optreden, worden binnen eenzelfde toestand toch veel waarnemingen gedaan en is alles wel OK. Maar dat is, zoals we eigenlijk zojuist opmerkten, ook mogelijk als de frequentie van samplen van dezelfde orde van grootte is als de snelheid waarmee de toestanden die gemeten moeten worden onderling stuivertje wisselen. Of als de samplingfrequentie duidelijk lager ligt. Mits maar, en dat is essentieel, de momenten van meten, van samplen, helemaal niet gecorreleerd zijn met de momenten van toestandswisseling. De momenten van sampling mogen inderdaad best periodiek zijn, als dan maar de toestandswisselingen geen enkele zuivere periodiciteit van die vorm vertonen.

Een voorbeeld is de meting van de CPU-tijd van programma's onder UNIX. Van ieder proces kan met het commando 'time' onder andere de verbruikte CPU-tijd worden opgevraagd. Deze tijd wordt op de volgende manier verzameld. Iedere clocktick geeft de clock een interrupt. Bij een netfrequentie van 50 Hz zijn er 50 clockticken per sekonde en duurt een clocktick precies 20 msek. In de

clockinterrupt-handler wordt de systeemadministratie bijgewerkt, de prioriteiten worden bijvoorbeeld bijgesteld en er wordt gekeken wie de CPU heeft. Voor dat proces wordt een volle clocktick CPU-tijd genoteerd. De som van de genoteerde tijden wordt door time gemeld.

Er wordt dus verre van continu de CPU-tijd gemeten, en de frequentie van het periodieke samplen is lager dan de snelheid waarmee processen elkaar van de CPU verdringen. Toch zijn de gegevens van time redelijk betrouwbaar omdat er weinig processen met de clock in de pas lopen, of met een veelvoud van de clockticken correleren.

Helaas treedt de correlatie soms wel op tussen een programmeerbare clock en de (systeem)clock, die immers beide met vaste tussenpozen kijken. Deze complicatie maakt dat het meten met een programmeerbare clock wel eens lastiger is dan verwacht.

7.7. VARIATIECOEFFICIENT

We gebruikten het statistische begrip kansverdeling om het verschil in de toestandsduren te beschrijven. Belangrijke karakteristieken van een verdeling zijn verwachtingswaarde en variantie, de variantie is een maat voor de spreiding. In plaats van variantie of standaarddeviatie wordt vaak de dimensieloze grootheid variatiecoëfficiënt gehanteerd. Omdat deze coëfficiënt in belangrijke uitdrukkingen gaat optreden —en daardoor vaak de verdeling van de duren globaal karakteriseert— staan we stil bij deze klassieke statistische grootheid.

De variatiecoëfficiënt van een verdeling wordt gedefinieerd als de verhouding tussen standaarddeviatie en verwachtingswaarde. Toegespitst op de duur is dat

$$\text{variatiecoefficient(duur)} = \text{standaarddeviatie(duur)} / E(duur)$$

of

$$\text{variatiecoefficient}^2\text{(duur)} = VAR(duur) / (E(duur))^2$$

De dimensieloze variatiecoëfficiënt wordt meestal geschreven als C, waarbij

$$C^2 = VAR(duur) / E^2(duur)$$

De variatiecoëfficiënt geeft aan hoe groot de relatieve spreiding in de duren is. Als er precies één duur mogelijk is, er is dan een vaste duur, is er geen spreiding; de variantie en de variatiecoëfficiënt C zijn nul.

7.7.1. *indeling naar relatieve spreiding*

Verdelingen met een variatiecoëfficiënt kleiner dan 1 heten *hypo-exponentiële verdelingen*, hoewel die naam ook soms gereserveerd wordt voor de verdelingen met een variatiecoëfficiënt kleiner dan 1, die van het Coxiaanse type zijn (Coxian is een bepaalde mathematische specificatie, praktisch betekent het 'fatsoenlijk'). We zullen ons niet om dit onderscheid bekommeren. De hypo-exponentiële verdelingen hebben een relatief geringe spreiding.

Verdelingen met een variatiecoëfficiënt groter dan 1 heten *hyper-exponentiële* verdelingen. Bij dit soort verdelingen lopen de duren sterk uiteen: er zijn zowel nogal wat relatief korte naast nogal wat relatief lange duren. De hyper-exponentiële verdelingen hebben een relatief grote spreiding.

naam	variatie-coefficient	spreiding
hyper-exponentieel	>1	vrij groot
hypo-exponentieel	<1	vrij klein
o.a. negatief exponentieel	=1	gewoon

Tabel 7.9. Indeling verdelingen naar relatieve grootte van de spreiding.

Verdelingen met een variatiecoëfficiënt van 1, waarvoor dus standaarddeviatie en verwachtingswaarde even groot zijn, zitten tussen beide in. De fameuze negatief exponentiële verdeling hoort in deze groep. In het hoofdstuk SIGNALEN EN POISSONPROCES (§12.3.5) zullen we deze intrigerende verdeling uitgebreid bekijken. Men zou kunnen zeggen dat verdelingen met een variatiecoëfficiënt van 1 een 'gewone' spreiding hebben.

7.7.2. *voorbeelden relatieve spreiding*

De verdeling van de verwerkingsduren in het voorbeeld DATA, CODE, KERNEL (§7.2.4) heeft een gemiddelde waarde van 3/40 msek. en een variantie (zie de volgende subparagraaf) van 23/12800 (msek.)^2. Dus is het kwadraat van de variatiecoëfficiënt

$$C^2 = (23/12800)/(3/40)^2 \qquad = 23/72$$

Het is dus een hypo-exponentiële verdeling.

De verdeling in het voorbeeld BERICHTEN EN ACK'S (§7.3.4; tabel 7.6) is uniform. We bepaalden de verwachtingswaarde en de variantie. Een uniforme verdeling

over een interval *a-b*, $b > a$ heeft een variatiecoëfficiënt waarvoor

$$C^2 = \frac{(b-a)^2/12}{((b+a)/2)^2} = 1/3\frac{(b-a)^2}{(b+a)^2}$$

Deze is altijd <1: een hypo (a≥0); deze uniforme verdeling heeft dus een relatief kleine spreiding. De verdeling van de transfertijden heeft een variatiecoëfficiënt van $3/11\ \sqrt{3}$.

Bij de bekende normale verdeling zijn de verwachtingswaarde en de standaard-deviatie onafhankelijk van elkaar. Bij normaal verdeelde duren kunnen dan ook zowel relatief sterk gespreide als gering gespreide verdelingen optreden.

De verwerkingsduur door een disk bestaat uit drie gedeelten: de seektijd nodig om de kop te positioneren, de rotational delay totdat de juiste sector onder de kop doordraait en de transfertijd, waarin de informatie bitsgewijs over het toegangs-kanaal binnenkomt vanaf de schijf of waarin de informatie op de schijf wordt gezet (§8.9). Bij praktijkmetingen blijkt dat de som van deze duren een vrij geringe spreiding heeft: de verwerkingsduur door het randapparaat heeft een hypo-exponentiële verdeling. Dit komt omdat er geen erg grote verschillen zitten in de mogelijke seektijden, de rotational delay ongeveer uniform verdeeld zal zijn en de transfers meestal in vaste porties (pagina's, blokken) plaats vinden, met weinig spreiding in transfertijd.

De totale verwerkingsduren bij de processor zijn daarentegen nogal eens hyper-exponentieel verdeeld. Er zijn naast lange (batchachtige) jobs, die veel executietijd vragen, korte interaktieve jobs die bijna direkt klaar zijn.

7.7.3. vervolg voorbeeld: data, code, kernel; variantie (7.2.4)

De variantie in de verwerkingsduur is

$$VAR(verwerkingsduur) =$$

$$1/5\times(1/16-3/40)^2+2/5\times(1/8-3/40)^2+2/5\times(1/32-3/40)^2$$

of

$$VAR(verwerkingsduur) = E(duur^2)-E^2(duur) =$$

$$1/5\times(1/16)^2+2/5\times(1/8)^2+2/5\times(1/32)^2-(3/40)^2$$

Hoe ook bepaald, $VAR(verwerkingduur) = 23/12800$ msek.^{2}

7.8. TIJDGEMIDDELDE, RELATIE VAN LITTLE

In het hoofdstuk INLEIDING OPERATIONELE ANALYSE kwam een beetje terloops direkt al de operationele schatter $'p_{toestand}$, of $'p_Z$, naar voren (§3.4). We hebben inmiddels een veel duidelijker beeld van wat de aan deze schatter gerelateerde kans inhoudt. Zowel p_Z als b_Z zijn belangrijke grootheden, maar ze stellen essentieel verschillende kansen voor. We kunnen ons zo langzamerhand wel indenken waarom p_Z ter onderscheiding ook wordt aangeduid als de *tijdgemiddelde* kans op de beurt. In de operationele schatter voor p_Z staat eigenlijk dat bij p_Z over de tijd wordt gemiddeld, het is een tijdgemiddelde. Bij b_Z wordt over 'aantallen' gemiddeld, dat zou een 'aantal-gemiddelde' genoemd kunnen worden.

In een preciezere mathematische formulering is het zo: als de echte kans op aantreffen trendmatig verandert met de tijd t volgens $q_Z(t)$ is

$$p_Z = \int dt\, q_Z(t) / T$$

waarin de integraal loopt over een interval van lengte T. Op de nette formulering van de tijdmiddeling gaan we niet verder in.

Het is uiterst zinvol vast te stellen dat in de visie van de 'lukrake waarnemer' een impliciete middeling over het gebeuren in de tijd is aangebracht. Dit doet zich al voor bij de relatie van Little (§4.5). Ook het 'gemiddelde aantal' uit deze relatie is een 'tijdgemiddelde', en het is tevens het aantal dat gemiddeld door een waarnemer zal worden gevonden, die op lukrake momenten waarneemt. De lukrake waarnemer is, zo te zeggen, een middelaar over het tijdgebeuren. Zijn gezichtspunt is bij vele globale uitspraken relevant.

De relatie van Little is dus een betrekking tussen het tijdgemiddelde aantal en de gemiddelde duur en stroom. Ook het gemiddelde aantal terminalisten in de responsetijdrelaties is een tijdgemiddelde, het is het aantal dat een lukrake waarnemer gemiddeld zou vinden. Zo hebben we —als bijprodukt van dit hoofdstuk— nog wat preciezer geformuleerd wat de grootheden uit de voorgaande hoofdstukken inhouden. Het verschil tussen p_Z en b_Z legt dus een universeel onderscheid bloot, vandaar dat we er zo lang bij stilstonden.

Bij de responsetijdrelaties zagen we in de toepassingen (§5.8) heel duidelijk hoe sterk het relatieve aandeel van een bepaald soort werk in de 'gemiddelde stroom' kan verschillen van het relatieve aandeel er van in de 'gemiddelde aantallen'. Eigenlijk gaat het daar ook al om hetzelfde onderscheid, in het gemiddelde aantal telt via Little ook de duur mee.

7.9. SAMENVATTING

In dit hoofdstuk hebben we geen schokkende dingen geleerd, maar we bekeken wel van alle kanten de nuttige relatie tussen optreden en aantreffen, tussen kansen op

de beurt en tijdgemiddelde kansen:

kans op aantreffen met de beurt

$$= \textit{kans op optreden} \; \frac{\textit{gemiddelde duur van de beurt}}{\textit{gemiddelde overall duur van de beurt}}$$

Of in formule

$$p_Z = b_Z \frac{E(duur_Z)}{E(duur)}$$

Deze basisrelatie geldt zowel tussen de echte gemiddelden, dus de verwachtingswaarden, als tussen de gemeten gemiddelden; we zullen er voortaan naar refereren als de basisrelatie voor de kansen op optreden en aantreffen. De basisrelatie voor de duren valt er ook onder.

Deze basisrelatie is een betrekking —net als de relatie van Little— die iedereen in voorkomende gevallen na enige aarzeling zeker zal toepassen, ook als hij de voorgaande filosofieën niet kent. Het voornaamste doel van onze bespreking is eigenlijk om deze aarzeling weg te nemen. Bovendien verhoogt het gebruik van deze 'wetten' de herkenbaarheid en de leesbaarheid van redeneringen, en daarmee de overtuigingskracht en de communicatie met anderen. Denk eens in hoe moeilijk het zou zijn om de eenvoudigste electrotechnische redeneringen over te brengen als de wet van Ohm niet was geformuleerd.

7.10. OPGAVEN

opgave 7.1: vervolg gebundelde Remote-write's (7.3.5)

De verdeling van de duren van Remote-write operaties is als volgt. Remote-write-1 duurt in 1/3 van de gevallen 10 msek. en anders 40 msek. Remote-write-2 duurt steeds 40 msek. Een Remote-write-3 operatie duurt in 3 van de 4 gevallen 40 msek., in 1 van de 8 gevallen 70 msek. en in 1 van de 8 gevallen 90 msek. (de opgegeven waarden zijn de precieze waarden, er is geen verdere spreiding).

- Geef de verdeling van de duren van de Remote-write operaties. Is de verdeling hypo-exponentieel?
- Geef de kansen op het aantreffen van de mogelijke duren.

opgave 7.2: interrupt-monitoring (12.5.5)

De centrale verwerkingseenheid (CPU) kan interrupts van randapparaten opvangen. De CPU verwerkt deze interrupts in een interrupt-afwikkelingsroutine. Er zijn 3 soorten interrupts. Van alle gegenereerde interrupts is 1/6 deel van soort I, de verwerkingsduur van deze interrupts (duur van de routine) is gemiddeld 30 msek. Verder is 1/3 deel van soort II met een duur van gemiddeld 15 msek. en de helft van soort III met een duur van gemiddeld 20 msek. Er is geconstateerd dat de CPU bezig is met het afwikkelen van interrupts. Men wil graag weten met welke interrupt de CPU bezig is en zendt daarom een hardware-signaal dat op een lukraak moment constateert met welke interrupt-afwikkelingsroutine de CPU bezig is. Neem eerst aan dat de verschillende verwerkingsduren exact 30, 15 en 20 msek. zijn.

- Geef aan hoe groot de kans is dat de CPU met de interrupt-routine voor soort I bezig blijkt.
- Wat is het antwoord op de voorgaande vraag als niet bekend is dat de verwerkingsduren precies 30, 15 en 20 msek. zijn?
- En als niet bekend is dat de CPU met een interrupt bezig is?
- Is de verdeling van de verwerkingsduren hyper-exponentieel?

8

Restduren en wachtduren

8.1. INLEIDING

8.1.1. restduren

Een bepaalde toestand duurt in het algemeen niet steeds even lang, de duren hebben een zekere verdeling. Als deze toestandsduren worden waargenomen door een lukrake waarnemer treft hij echter deze verdeling niet aan! Wanneer bijvoorbeeld de toestandsduren uniform of negatief exponentieel verdeeld zijn, vindt een lukrake waarnemer geen uniforme of negatief exponentieel verdeelde toestandsduren.

Dat komt niet omdat de waarneming het systeem beïnvloedt, maar omdat een langere duur meer kans loopt om aangetroffen te worden dan een kortere.

Een aangetroffen toestandsduur is na enige tijd voorbij, er valt nog in de *restduur* van te genieten. Restduren spelen een belangrijke rol in de praktische computer-prestatieanalyse. Een verdeling van duren wordt meestal gezien door het 'filter' van de restduren. Het verband tussen de gemiddelde restduur van de door een lukrake waarnemer aangetroffen toestand en de verdeling van de toestandsduren is eenvoudig:

$$E(restduur) = \frac{E(duur^2)}{2E(duur)}$$

8.1.2. wachtduren

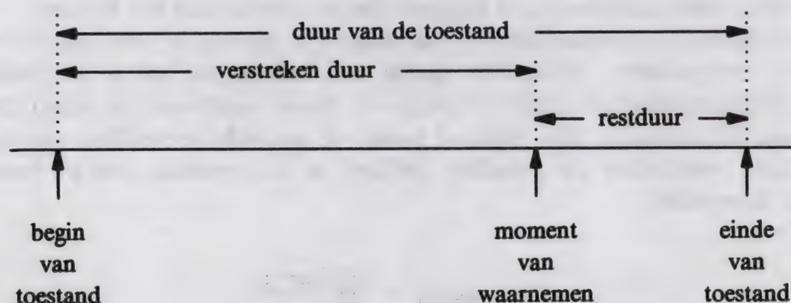
In veel situaties wordt er gewacht tot een toestand voorbij is en is een restduur ook een *wachtduur*. We bepalen de gemiddelde wachtduur voor de gevallen waarin de totale wachtduur uit deze restduur bestaat. De gemiddelde wachtduur als er op anderen moet worden gewacht, staat aangegeven in het hoofdstuk HET M/G/1 SYSTEEM.

De randapparaten van moderne grote mainframes vormen samen een complexe 'I/O-configuratie'. Het pad tussen CPU en disk loopt over channel (kanaal), storage director en controller. Op deze I/O-configuratie passen we de redeneringen rond de 'kansen op aantreffen' toe. We vinden een uitdrukking voor de gemiddelde responsetijd voor een diskopdracht.

8.2. FORMULE GEMIDDELDE RESTDUUR

Het komt vaak voor dat moet worden aangegeven hoelang het duurt tot de volgende toestandswisseling plaats vindt. Er wordt niet gevraagd naar de toestand waarin het systeem verkeert, maar tot wanneer het systeem nog in deze toestand zal blijven. We noemen dit de vraag naar de *restduur*, de huidige toestand blijft nog gedurende de restduur heersend.

We gaan met behulp van de basisrelatie voor de kansen op het optreden en aantreffen van duren uit het voorgaande hoofdstuk bepalen hoe lang de restduur gemiddeld zal zijn, wanneer de toestand van het systeem op lukrake momenten wordt geobserveerd (figuur 8.1).



Figuur 8.1. Restduur.

We zijn gespitst op de duren van toestanden. Als bij een lukrake waarneming $toestand_i$ van een zekere duur $duur_i$ wordt aangetroffen, duurt het gemiddeld nog $duur_i / 2$ voordat deze $toestand_i$ overgaat in een andere. (Dit is waarschijnlijk 'van zelf sprekend', de overweging daarbij zal zijn: Als we lukraak kijken moeten de

momenten waarop we blijken te kijken 'evenredig' (uniform) verdeeld zijn over de duur van de toestand (§13.2). De afstand tot het eindpunt is nu eens vrijwel de volle duur $duur_i$ en dan weer heel kort, maar gemiddeld volgt nog de halve toestandsduur $duur_i$. Als we een toestand van duur $duur_i$ aantreffen zal de restduur daarom gemiddeld $duur_i / 2$ zijn, want dat is het gemiddelde van een uniforme verdeling tussen 0 en $duur_i$ (§7.3.4).)

Maar we treffen niet altijd een toestand van duur $duur_i$ aan, zo'n toestand wordt met de kans p_i waargenomen. Gemiddeld over alle mogelijkerwijs aan te treffen duren (toestanden) vinden we als gemiddelde restduur (bij lukrake waarneming —maar dat vermelden we maar niet meer steeds)

$$E(restduur) = \sum_i p_i \; duur_i / 2$$

waarin de som loopt over alle mogelijkheden.

Deze uitdrukking gaan we herschrijven, we willen b_i in plaats van p_i hanteren om op verwachtingswaarden uit te komen. Invullen van de basisrelatie voor de kansen op het optreden en aantreffen van duren uit het hoofdstuk BEURTEN EN KANSEN (§7.2.3):

$$p_i = b_i \, duur_i / E(duur)$$

geeft

$$E(restduur) = \sum_i b_i \; duur_i^2 / (2E(duur))$$

In deze uitdrukking herkennen we inderdaad een verwachtingswaarde en wel van het kwadraat van de duur, dus er staat

$$E(restduur) = \frac{E(duur^2)}{2E(duur)} \tag{8.1}$$

De gemiddelde restduur wordt via deze formule bepaald uit de verdeling van de toestandsduren. De gevonden betrekking is natuurlijk zowel bij een discrete als bij een continue verdeling over de duren geldig. Restduren heten in het Engels 'residual times'. De betrekking is een fameus standaardresultaat voor de 'expected residual lifetime' in de renewal theorie (§13.4).

De uitdrukking voor $E(restduur)$ is opmerkelijk. Van de verdeling over de toestandsduren hoeft alleen maar de verwachtingswaarde van de duur en van het kwadraat van de duur bekend te zijn, meer informatie is niet nodig om de verwachtingswaarde van de restduur aan te geven!

In plaats van $E(duur^2)$ —het tweede moment van de verdeling— kan ook de variantie $VAR(duur)$ worden gehanteerd (§7.7.3). Via

$$VAR(duur) = E(duur^2) - E^2(duur)$$

volgt

$$E(restduur) = E(duur)/2 + VAR(duur)/(2E(duur)) \tag{8.2}$$

Hier staat dat de gemiddelde restduur niet gelijk is aan de halve gemiddelde toestandsduur $E(duur)/2$, nee, er komt altijd als extra bijdrage een positieve term $VAR(duur)/(2E(duur))$ bij. Het extra stuk is evenredig met de variantie. Alleen als de spreiding, dus $VAR(duur)$, nul is, is de gemiddelde restduur gelijk aan de halve duur. Hoe groter de spreiding in de duren, hoe groter de extra term en hoe meer verschil er is tussen de gemiddelde restduur en de halve gemiddelde toestandsduur.

Als de spreiding nul is hebben alle toestanden dezelfde duur en wordt er altijd dezelfde duur gevonden. Alleen dan is

$$E(restduur) = E(duur)/2$$

We zijn consistent, in de afleiding brachten we rechtstreeks in dat dit het geval is.

De eenvoudige reden van het verschil tussen $E(restduur)$ en $E(duur)/2$ is opnieuw dat de kansen op optreden en op aantreffen niet hetzelfde zijn. Het is simpelweg zo, dat er bij spreiding in de duren een grotere kans is om in een langere duur terecht te komen (te 'prikken'), dan om een kortere duur te treffen. En daardoor komen langere restduren vaker voor dan volgens de kansen op optreden wordt aangegeven.

Als we de variatiecoëfficiënt C van de duur invoeren krijgt de formule voor de gemiddelde restduur de vorm (§7.7)

$$E(restduur) = E(duur)/2 + C^2 E(duur)/2$$

of

$$E(restduur) = (1+C^2)E(duur)/2 \tag{8.3}$$

8.2.1. gemiddelde restduur en variatiecoefficient

Bij een hypo-exponentiële verdeling over de duren is $C<1$ (zie §7.7, tabel 7.9). Voor die verdelingen is de extra bijdrage kleiner dan de halve gemiddelde duur, zodat de gemiddelde restduur kleiner is dan de gemiddelde duur. Er moet als er betrekkelijk weinig spreiding is in de toestandsduren, gemiddeld korter dan de gemiddelde toestandsduur worden gewacht op een toestandsverandering.

Bij toestandsduren die hyper-exponentieel verdeeld zijn is $C>1$ en is de extra bijdrage groter dan de halve gemiddelde duur. Nu is de gemiddelde restduur groter dan de gemiddelde duur. Er moet als er vrij veel spreiding zit in de toestandsduren gemiddeld langer dan de gemiddelde toestandsduur worden gewacht op een toestandsverandering —hoe paradoxaal dat ook klinkt.

Verdelingen met een variatiecoëfficiënt $C=1$, waarvoor standaarddeviatie en verwachtingswaarde even groot zijn, zitten tussen beide in; voor hen is de gemiddelde restduur even lang als de gemiddelde duur. Vanaf een willekeurig moment moet dan gemiddeld nog de gemiddelde toestandsduur gewacht worden op toestandsverandering.

Voor negatief exponentieel verdeelde toestandsduren is de variatiecoëfficiënt 1 en duurt het vanaf elk, hier zelfs niet per se lukraak genomen moment, nog gemiddeld de gemiddelde toestandsduur voordat de toestand voorbij is (we zullen dit zien in het hoofdstuk SIGNALEN EN POISSONPROCES en §13.2).

In de uitkomst (8.1) voor de gemiddelde restduur treedt de uitdrukking $E(duur^2)$ op. Helaas wordt er bij het hanteren van deze uitkomst nogal eens rücksichtslos voor deze uitdrukking het kwadraat van de gemiddelde duur genomen. Dat is natuurlijk wezenlijk fout, tenzij er geen spreiding zit in de duren. De grootte van de fout is precies de variantie.

8.3. SCHEMA BEPALING GEMIDDELDE RESTDUUR

Voor het bepalen van de gemiddelde restduur is het dus niet nodig eerst de hele verdeling van de duren op te stellen, bepaling van $E(duur^2)$ is genoeg. Er zijn daarvoor korte schema's, zoals het volgende *'recurrente'*, waarin zelfs niet $E(duur^2)$ wordt bijgehouden. Zo'n schema kan worden gebruikt om snel even 'op de achterkant van een envelop' de gemiddelde restduur uit te rekenen.

Veronderstel dat in eerste instantie een aantal (hoofd)toestanden worden onderscheiden, deze geven we aan met (de index) A. Als het systeem in $toestand_A$ wordt aangetroffen, duurt het gemiddeld nog $E(restduur_A)$ voordat die toestand voorbij is. De 'overall' gemiddelde restduur is, zoals uit de afleiding in §8.2 blijkt (de som is over de toestanden A)

$$E(restduur) = \sum_A p_A \, E(restduur_A) \tag{8.4}$$

Als $toestand_A$ wordt aangetroffen beperkt dat de mogelijke restduren tot de

restduren binnen die toestand. Voor $E(restduur_A)$ kunnen we een van de gelijkwaardige uitdrukkingen (8.1), (8.2) of (8.3) gebruiken. Met (8.3) is

$$E(restduur_A) = E(duur_A)(1+C_A^2)/2 \tag{8.5}$$

Dit gebruiken we als we de grootheden in het rechterlid kennen, dus als het gemiddelde en de variatiecoëfficiënt (of de variantie, of het tweede moment) van de duur van $toestand_A$ bekend zijn. Anders gaan we rustig 'recursief' verder door $toestand_A$ in zijn (sub)toestanden te splitsen, waarmee als in (8.4) (de som is over de subtoestanden AB van A)

$$E(restduur_A) = \sum_{AB} p_{AB}\ E(restduur_{AB}) \tag{8.6}$$

$Toestand_A$ wordt met een kans p_{AB} aangetroffen als $toestand_{AB}$ met gemiddelde restduur $E(restduur_{AB})$.

Vervolgens wordt $E(restduur_{AB})$ volgens hetzelfde patroon bepaald, enzovoort. We gaan ermee door tot we uitkomen bij toestanden met een bekende gemiddelde duur en een bekende variatiecoëfficiënt voor de duur. Veel voorkomende gevallen zijn de vaste duur ($C=0$), waarbij de gemiddelde restduur gelijk is aan de halve gemiddelde duur en de negatief exponentieel verdeelde duur, waarvoor de gemiddelde restduur gelijk is aan de gemiddelde duur ($C=1$, §8.2.1). In de praktijk bij verdelingen, die als benadering van praktische situaties dienst doen, komen we al heel gauw alleen nog maar zulke eenvoudige restduren tegen en hoeven we nooit 'diep' in het schema te gaan.

We passen dit schema toe op het voorbeeld 'spreiding in duren' uit het hoofdstuk BEURTEN EN KANSEN; de relevante gegevens uit tabel 7.5 zijn overgenomen in tabel 8.1. Er treden duren op van 10, 15, 20, 30 en 80 met kansen 1/2, 1/9, 5/18, 1/18 en 1/18. De gemiddelde duur is 55/3. Het gemiddelde kwadraat van de duur is

$$E(duur^2) = 10^2/2+15^2/9+5\times20^2/18+30^2/18+80^2/18 = 5325/9$$

zodat

$$E(restduur) = (5325/9)/(2\times55/3) = 355/22 = 16\frac{3}{22}$$

Met het 'schema' loopt het zo: De kansen om de toestanden A, B en Z aan te treffen bepaalden we op 2/11, 6/11 en 3/11. Dus is volgens (8.4)

$$E(restduur) = 2/11\,E(restduur_A)+6/11\,E(restduur_B)+3/11\,E(restduur_Z)$$

	toestanden					
	A		*B*		*Z*	
gem. duur	*20*		*30*		*10*	
kans	1/6		1/3		1/2	
	optreden					*gem. duur*
duur (vast)	15	30	20	80	10	
optreden	2/3	1/3	5/6	1/6	1	
kans	1/9	1/18	5/18	1/18	1/2	55/3
	aantreffen					
kans	1/11	1/11	10/33	8/33		
	2/11		6/11		3/11	
	restduur					
gemiddeld	15/2	15	10	40		355/22
	45/4		70/3		5	355/22

Tabel 8.1. Gemiddelde restduren 'spreiding in duren' (tabel 7.5).

Als toestand A wordt aangetroffen, is er een kans van $(1/3\times30)/20 = 1/2$ dat de duur 30 wordt gevonden, evenzo is er een kans van 1/2 dat de duur 15 zich voordoet. De gemiddelde restduur als A heersend blijkt te zijn is volgens (8.6)

$$E(restduur_A) = 1/2\times15/2 + 1/2\times30/2 = 45/4$$

Als toestand B wordt aangetroffen is er een kans van $5/6\times20/30 = 5/9$ dat de toestand 20 duurt, idem 4/9 dat hij 80 duurt:

$$E(restduur_B) = 5/9\times20/2 + 4/9\times80/2 = 70/3$$

Toestand Z duurt precies 10 t.e., dus is $E(restduur_Z) = 5$. Samen levert dit op

$$E(restduur) = 2/11\times45/4 + 6/11\times70/3 + 3/11\times5$$

en dat is weer 355/22.

De beide methoden leveren altijd dezelfde waarde voor de gemiddelde restduur, zoals blijkt door alle deeluitdrukkingen uit het 'schema' uit te schrijven, het zijn gewoon subtermen van $E(duur^2)$.

8.3.1. *samenvatting*

De uitdrukking voor de gemiddelde restduur is te schrijven in de vormen

$$E(restduur) = \frac{E(duur^2)}{2E(duur)} \qquad (8.1)$$

$$E(restduur) = E(duur)/2 + VAR(duur)/2E(duur) \qquad (8.2)$$

$$E(restduur) = (1+C^2)E(duur)/2 \qquad (8.3)$$

$$E(restduur) = \sum_A p_A \, E(restduur_A) \qquad (8.4)$$

Voor de gemiddelde restduur $E(restduur_A)$ van de mogelijke toestanden A kan opnieuw (8.1) tot en met (8.4) worden gebruikt.

8.4. TOEPASSINGEN

8.4.1. *voorbeeld: resterende executieduur*

De CPU-tijden van programma's zijn gemiddeld 20 msek. aan usertijd en systeemtijd samen. De verdeling van deze tijden is hyper-exponentieel met een variatiecoëfficiënt van 3. Deze gegevens zijn bijvoorbeeld te verkrijgen met een faciliteit als 'time', die in UNIX rapporteert over de verbruikte CPU-tijd.

Op een bepaald moment wordt gevraagd hoeveel CPU-tijd het thans executerende programma nog gemiddeld zal verbruiken. Het antwoord (volgens (8.3), thans is een lukraak moment) is $20/2 \times (1+3^2) = 100$ msek. en niet 10 msek. Deze tijd zal het programma niet in één keer opvragen waarschijnlijk, maar —afhankelijk van de scheduling— in partjes.

8.4.2. *vervolg voorbeeld: schema*

Toestand A treedt met een kans van 1/3 op als een toestand, zeg Ab, met een duur van precies 30 (zie tabel 8.1). Als deze toestand Ab niet een vaste duur heeft, maar een duur met variatiecoëfficiënt 2, verandert er weinig in de berekening volgens het schema voor de gemiddelde restduur. Alleen $E(restduur_A)$ wordt

$$E(restduur_A) = 1/2 \times 15/2 + 1/2 \times 30/2 \times (1+4) = 165/4$$

Dus dan is $E(restduur) = 355/22 + 1/11 \times 30/2 \times 4 = 21\,\frac{13}{22}$

8.4.3. vervolg voorbeeld: pollen (7.5.4; tabel 7.8)

De in het vorige hoofdstuk gevonden uitkomsten voor het voorbeeld 'pollen' (§7.5.4) nemen we over in tabel 8.2.

	pollen			
gem. duur (/ $duur_{poll}$)	transfer 4	transfer-item 4/3	poll 1	
	optreden			*gem. duur*
kans	6/13	18/25	7/13 7/25	31/13
	aantreffen			
kans	24/31	24/31	7/31	
	gemiddelde restduren			
(duren vast)	2	2/3	1/2 1/2	103/62 39/62

Tabel 8.2. Gemiddelde restduren pollen.

We vergelijken de gemiddelde restduren, waarbij we aannemen dat de toestandsduur van toestand 'poll' steeds dezelfde duur $duur_{poll}$ is en die van toestand 'transfer' steeds $4duur_{poll}$. We gaan er eerst van uit dat de toestand voorbij is als alle subtoestanden zijn afgelopen. De gemiddelde restduur als toestand 'transfer' wordt aangetroffen is dan $4duur_{poll}/2 = 2\,duur_{poll}$. De 'overall' gemiddelde restduur is volgens (8.4)

$$(2p_{transfer} + 1/2\,p_{poll})\,duur_{poll} = (48/31 + 7/62)\;duur_{poll} = 1\frac{41}{62}\;duur_{poll}$$

Dit volgt ook uit de vorm (8.1):

$$E(duur) = (7/13\times 1 \;+\; 6/13\times 4)\times duur_{poll} = 31/13\;duur_{poll}$$

$$E(duur^2) = (7/13\times 1 \;+\; 6/13\times 16)\times duur_{poll} = 103/13\;duur^2_{poll}$$

$$E(restduur) = E(duur^2)/(2E(duur)) = 103/62\;duur_{poll}$$

Als we de restduur nemen tot het einde van een subtoestand is de gemiddelde restduur, wanneer een transfer steeds uit 3 transferitems van dezelfde duur bestaat, niet 103/62 $duur_{poll}$, maar

$$(2/3 p_{transferitem} + 1/2 p_{poll})\, duur_{poll} = 39/62\ duur_{poll}$$

8.4.4. paradox aangetroffen duur

In plaats van naar de resterende duur van de aangetroffen toestand te vragen kunnen we ook informeren naar de reeds verstreken duur. Dat is de tijd dat de aangetroffen toestand al de 'heersende' is, dus de tijd verstreken tussen het moment waarop de aangetroffen toestand zijn voorganger verving en het moment waarop we kijken (figuur 8.1).

Op grond van symmetrie zien we direkt in dat de gemiddelde verstreken duur gelijk zal zijn aan de gemiddelde restduur, zolang we op lukrake momenten kijken. De som van verstreken duur en restduur is de aangetroffen duur. De gemiddelde waarde daarvan is

$$E(aangetroffen\ duur) = E(verstreken\ duur) + E(restduur) = 2\,E(restduur)$$

dus

$$E(aangetroffen\ duur) = E(duur^2)\,/\,E(duur)$$

Dit resultaat vonden we in §7.5.3 uit BEURTEN EN KANSEN rechtstreeks.

De verstreken tijd zal bij negatief exponentieel verdeelde duren net als de restduur gemiddeld gelijk zijn aan $E(duur)$. Dan is dus

$$E(aangetroffen\ duur) = 2E(duur)$$

Zijn de duren negatief exponentieel verdeeld rond een gemiddelde waarde van 6 sekonden, dan kom je binnen in toestanden die gemiddeld 12 sekonden duren! Die uitkomst is misschien niet direkt voor de hand liggend. Maar het is nu wel duidelijk dat er eigenlijk helemaal geen paradox is. Er is een essentieel verschil tussen het aantreffen en het optreden van een toestand. En hoe langer de duur van een toestand, hoe groter de kans dat je er in terecht komt.

Voor de gemiddelde restduur geldt, als generalisatie van de uitkomst van de halve duur bij een vaste duur,

$$E(restduur) = E(aangetroffen\ duur)\,/\,2$$

8.4.5. vervolg voorbeeld: berichten en ack's (7.3.4; tabel 7.6; 7.7.2)

Op een lukraak moment treft men op de lijn een bitstroom uit een bericht of uit een acknowledgement aan; er loopt een 'boodschap', die met 30 Kbits/sek. overgestuurd wordt. De aangetroffen duur is òf een bericht met een uniform verdeelde duur tussen 10 en 100 msek. òf een acknowledgement met een duur van precies 5.5 msek. (laten we aannemen dat ack's een vaste lengte hebben). De in §7.3.4 berekende kansen en de variatiecoëfficiënt uit §7.7.2 verzamelen we in tabel 8.3.

	berichten en ack's				
type	gem. duur	C^2	kans op optreden	kans op aantreffen	gem. duur aangetroffen
bericht	55	27/121	10/11	100/101	
ack	5.5	0	1/11	1/101	
overall	50.5				66.7

Tabel 8.3. Berichten en acknowledgements.

De transfertijd van de aangetroffen boodschap is gemiddeld het dubbele van de gemiddelde restduur, dus volgens (8.4)

$$\textit{gem. aangetroffen transf.} = 1/101 \times 5.5 + 100/101 \times 55 \times (1 + 27/121) = 66.7 \text{ msek.}$$

De gemiddelde transfertijd van de boodschappen is daarentegen

$$\textit{gem. transfertijd} = 1/11 \times 5.5 + 10/11 \times 55 = 50.5 \text{ msek.}$$

8.4.6. voorbeeld: wachten-op-de-bus

Je benut het openbaar vervoer bij al je verplaatsingen. Om het station te bereiken neem je de stadsbus. Op de dienstregeling in de abri heb je gelezen dat de bussen om de 20 minuten langs komen, maar wanneer ze precies voorrijden heb je niet kunnen onthouden. Als je je dan naar de bushalte begeeft, kom je daar aan op een lukraak moment 'tussen twee bussen'. Omdat je totaal vergeten bent wanneer precies de bus langs komt is er geen correlatie tussen de momenten, waarop jij er aan komt lopen, en de momenten waarop de bus er aan komt.

Goed, je staat in de abri. Hoelang duurt het gemiddeld nog totdat de bus komt? Het antwoord is 10 minuten als de bussen precies om de 20 minuten langs komen. Maar als de bussen niet precies om de 20 minuten komen? Als de chauffeurs proberen de service te handhaven komen ze wel gemiddeld om de 20 minuten langs. Wanneer er een bus voorbij is en 18 minuten daarna komt de

volgende, dan heeft een toestand met duur 18 de beurt gehad. Komt de volgende bus 25 minuten later dan kreeg daarna de duur 25 de beurt. De vraag naar de tijd, die nog gewacht moet worden op de volgende bus, kan kennelijk vertaald worden als een vraag naar de restduur. En wel hier naar de restduur als op willekeurige momenten gekeken wordt. Daarvoor is de uitdrukking bekend: er moet gemiddeld nog gewacht worden

$$E(wachtduur) = E(tussenpoos\ bussen^2)/2E(tussenpoos\ bussen)$$

of

$$E(wachtduur) = E(tussenpoos\ bussen)/2 + idem \times C^2$$

met $E(tussenpoos\ bussen) = 20$ minuten.

Hoelang je moet wachten hangt er niet alleen van af om de hoeveel tijd de bus langs komt, maar ook van de spreiding in deze tijd. Worden de tussenpozen verkort, maar heeft dat tot gevolg dat de spreiding door opstoppingen en storingen toeneemt, dan betekent dat voor je gemiddelde wachttijd lang niet direkt voordeel. Bij grote spreiding in de tussenpozen moet je zelfs gemiddeld langer wachten dan de gemiddelde tussenpoos van 20 minuten!

In Amsterdam reed tot voor kort de tram inderdaad volgens het principe: als de gemiddelde tussenpozen maar kort zijn, is het wel goed. Maar de klachten van reizigers hebben resultaat opgeleverd; er zijn computers ingeschakeld om de loop regelmatiger te maken, de posities van de trams worden bijgehouden.

8.4.7. voorbeeld: Mean Time Between Failures

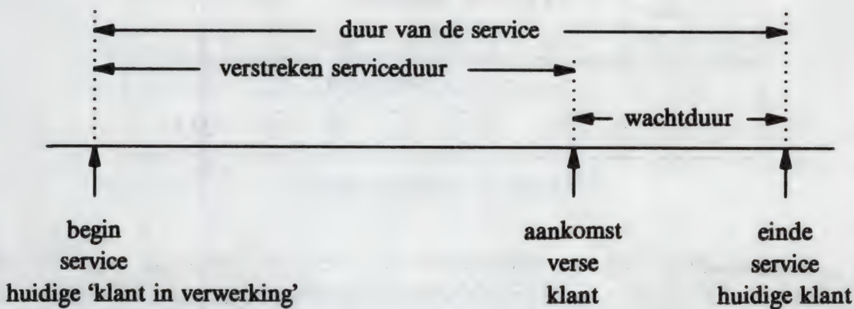
Bij apparatuur wordt gewoonlijk vastgelegd hoeveel tijd er verstrijkt voordat de volgende storing optreedt, de tijdintervallen tussen opeenvolgende storingen worden steeds genoteerd. Deze tussenpozen bij storingen hebben een verdeling die meestal nogal veel spreiding vertoont, storingen zijn vaak 'bursty': nu eens telkens weer, dan weer heel lang geen enkel probleem. De verwachtingswaarde van deze tussenpozen wordt de MEAN TIME BETWEEN FAILURES genoemd. Ieder rekencentrum geeft in zijn periodieke rapportages de MTBF voor de configuraties over de afgelopen periode.

Op een zeker moment meldt zich een verse klant voor het centrum, bijvoorbeeld een student die een opdracht moet uitvoeren, waarvoor hij ongeveer een week denkt nodig te hebben. Hij is niet ingewijd en weet niet wanneer de vorige storing was. Op maandagmorgen begint hij en krijgt te horen van een medeterminalist dat de MTBF ongeveer 1 week is. Moet hij schrikken? De gemiddelde restduur tot de komende storing zal langer zijn dan een week, omdat voor de verdeling van de intervallen tussen storingen de variatiecoëfficiënt $C > 1$ is. Als het rekencentrum uit waarnemingen heeft vastgesteld dat $C = 2$, zal het volgens (8.3) gemiddeld twee-en-een-halve week duren voor de volgende storing komt. Gelukkig is de kans vrij groot dat de student een lange tussenpoos treft!

8.5. RESTDUREN ALS WACHTDUREN

Restduren spelen ook een belangrijke rol als bestanddeel van wachtduren. Veel delen van computerconfiguraties gedragen zich als wachttijdensystemen; er wordt gewacht totdat de 'klant' door de server (vaak opnieuw) in behandeling wordt genomen.

We bekijken een 'non-preemptive' scheduling van klanten: het werk voor de één kan niet worden onderbroken voor een ander. Bij zulke regelingen moet er in ieder geval gewacht worden tot de server met zijn huidige 'klant' klaar is. We willen uitrekenen hoelang dit gemiddeld duurt. Deze wachttijd (wachtduur) is een restduur, de toestand is 'server is bezig met klant' en de verdeling van de duren van deze toestand is de verdeling van de serviceduren van de klanten. In de berekeningen voor de totale gemiddelde wachtduur in wachttijdensystemen en speciaal bij het M/G/1 systeem wordt nader gebruik gemaakt van deze bijdrage (zie §15.3).



Figuur 8.2. Restduur als wachtduur, tot server klaar is met huidige klant.

Het komt ook vaak voor dat de bedoelde restduur de totale wachttijd is. Er moet dan alleen gewacht worden tot de server gereed is. Dat is bijvoorbeeld het geval voor klanten van de hoogste prioriteit. Die hoeven —tenzij ze elkaar in de weg zitten— alleen te wachten als ze een klant bij de server treffen. Zodra de server klaar is, komen zij aan bod (vergelijk figuur 8.2 met figuur 8.1). We geven aanstonds een voorbeeld: wachten van een signaal in een I/O-configuratie.

Ook bij een 'batchgewijze' verwerking wordt er alleen gewacht totdat de server beschikbaar komt. Zo'n geval hadden we zojuist bij de bus: als de bus er is kan iedereen instappen, de hele 'batch' reizigers wordt in één keer opgenomen.

8.5.1. vervolg voorbeeld: gebundelde Remote-write's (7.3.5; tabel 7.7)

Zo'n geval is ook het wegschrijven van data bij de Remote-write requests uit het hoofdstuk BEURTEN EN KANSEN. We willen berekenen hoelang er gemiddeld gewacht wordt voordat de Remote-write request wordt gehonoreerd. Requests worden op lukrake momenten gegenereerd.

server	**Remote-write**			
request gem. duur verdeling	-1- 30 *neg.exp.*	-2- 40 *vast*	-3- 50 *normaal* *(C=1/5)*	
	kans op **optreden**			*gem. duur*
	0.83	0.13	0.04	32.1
idle	kans op **aantreffen**			
0.20	0.62	0.13	0.05	
	gemiddelde **restduren**			
	30	20	26	22.5

Tabel 8.4. Remote-write requests.

We veronderstellen dat er informatie is over de verdeling van de verwerkingsduren. De duren van Remote-write-1 operaties zijn negatief exponentieel verdeeld ($C=1$) met als gemiddelde 30 msek. Een Remote-write-2 operatie duurt steeds 40 msek. De Remote-write-3 operaties zijn normaal verdeeld rond het gemiddelde van 50 msek., met een standaarddeviatie van 10 msek. (dus $C=1/5$).

Een request dat gegenereerd wordt terwijl de server bezig is met een Remote-write-1 operatie moet gemiddeld wachten, volgens (8.3),

$$E(restduur_1) = 15\times(1+1) = 30 \text{ msek.}$$

Als het nieuwe request de toestand 'Remote-write-2' treft moet het gemiddeld wachten $E(restduur_2) = 40/2 = 20$ msek. Bij 'Remote-write-3' is $E(restduur_3) = 50/2\times(1+1/25) = 26$ msek.

We zagen in §7.3.5 dat een nieuwe request de Remote-write-1 operatie vindt met een kans van 0.62, Remote-write-2 met een kans van 0.13 en Remote-write-3 met een kans van 0.05. Met een kans van 0.20 is de server werkeloos en hoeft er niet te worden gewacht.

De gemiddelde wachtduur tot de server aan het oversturen begint is dus, volgens (8.4)

$$\textit{gem. wachtduur} = 0.20\times0+0.62\times30+0.13\times20+0.05\times26 = 22.5 \text{ msek.}$$

8.5.2. voorbeeld: vertraging signaal in een I/O-configuratie

In een I/O-configuratie wordt de communicatie tussen CPU, channels (kanalen), controllers en disks verzorgd door signalen (figuur 8.3 uit §8.8.1). Er is bijvoorbeeld geconstateerd dat een disk vrij is. Nu wordt er een signaal gestuurd vanaf de CPU naar de betrokken disk. Het pad waarlangs het signaal gaat blijkt dan helaas in 40% van de gevallen bezet te zijn. Het pad is soms niet vrij doordat er over een van de schakels uit het pad een transfer loopt, die gemiddeld 1.5 msek. duurt. Maar er kan ook een van de controllers uit het pad bezig zijn met zijn 'control protocol', dat gemiddeld 2.6 msek. duurt. We bepalen hoelang het signaal gemiddeld wordt opgehouden door het bezet zijn van het pad, als de beide mogelijkheden even vaak optreden.

Signalen hebben altijd de hoogste prioriteit, want ze moeten snel door en duren kort —veel korter dan 1.5 msek.— en geven dus weinig hinder, ook niet aan elkaar. De enige vertraging, die ze kunnen oplopen, is de pech een 'transfer' of een 'control protocol' aan te treffen. De gemiddelde wachtduur voor een signaal is de gemiddelde restduur voor de toestand 'bezet', waarbij bezet òf gemiddeld 1.5 msek. duurt òf gemiddeld 2.6 msek., met kansen $b_i = 1/2$. Laten we aannemen dat het signaal op een ten opzichte van het verkeer over het pad lukraak moment optreedt. En dat de duren precies 1.5 en 2.6 msek. lang zijn. De gemiddelde wachtduur is dan

$$\textit{gem. wachtduur} = 0.40\times(1.5/4.1\times1.5/2 + 2.6/4.1\times2.6/2) = 0.44 \text{ msek.}$$

Een signaal loopt gemiddeld een vertraging op van 0.44 msek.

8.6. M/G/1 (WACHTTIJDTHEORETISCH INTERMEZZO)

Het zou jammer zijn hier niet te vermelden wat de uitdrukkingen voor de gemiddelde wachtduur zijn in de bekendste 'open' wachttijdensystemen; dit zijn het M/G/1 en het M/M/1 systeem. We lopen daarmee vooruit op het hoofdstuk HET M/G/1 SYSTEEM, dus heel beknopt:
Een M/G/1 systeem is een wachttijdensysteem met 1 server en aankomsten volgens een Poissonproces (voor dit proces zie het hoofdstuk SIGNALEN EN POISSONPROCES). In zo'n systeem wordt niet alleen gewacht tijdens de zojuist telkens berekende 'wachttijd tot de klant die bij de server in verwerking is, is afgewerkt' (deze restduur noemen we ter onderscheiding de *wachtrestduur*), maar een klant

heeft nu ook last van anderen die voor of na hem binnenkomen en voor hem geholpen worden. Er wordt verondersteld dat bij de keuze van de volgende klant niet op de serviceduur wordt gelet.

De gemiddelde wachtduur zal dus langer zijn dan de gemiddelde wachtrestduur. Maar de gemiddelde extra bijdrage blijkt eenvoudig te beschrijven; hij wordt verder uitsluitend bepaald door de bezettingsgraad U van de server. De gemiddelde wachtduur in zo'n M/G/1 systeem is de gemiddelde wachtrestduur, vermenigvuldigd met de factor $1/(1-U)$. In formule vorm

$$gem.\ wachtduur = \frac{gem.\ wachtrestduur}{1-U}$$

Voor de teller kan de uitdrukking voor de gemiddelde restduur worden gebruikt, bijvoorbeeld (8.1). Deze moet vermenigvuldigd worden met U, omdat er een kans is van $1-U$ dat de server niet bezig is en er helemaal niet gewacht hoeft te worden. Er staat dus ook

$$gem.\ wachtduur = \frac{U \times E(serviceduur^2)/(2E(serviceduur))}{1-U}$$

Dit is de vermaarde formule van Pollaczek/Khinchin (§15.3).

Als de serviceduren van de klanten negatief exponentieel verdeeld zijn, is het M/G/1 systeem een M/M/1 systeem. De uitkomst voor de gemiddelde verblijfduur, dat is de som van de gemiddelde wachtduur en de gemiddelde serviceduur, luidt in zo'n systeem (§15.9):

$$gem.\ verblijfduur = \frac{gem.\ serviceduur}{1-U}$$

8.7. BUSY PERIOD

Een server (een processor uit de wachttijdentheorie) verwerkt verzoeken om hulp (bijvoorbeeld van processen). De server heeft af en toe niets te doen en dan weer gedurende een hele poos werk. Dit is een voorbeeld van de situatie bezet/vrij (§7.5.2): als de server bezig is met het verwerken van de verzoeken om hulp is hij bezet, als hij niets te doen heeft is hij vrij. Er geldt dus met U als bezettingsgraad van de server

$$U = E(continu)/(E(leegloop)+E(continu))$$

$E(leegloop)$ is de gemiddelde lengte van een tijdje nietsdoen en $E(continu)$ is de

gemiddelde duur van een periode van ononderbroken continu werken van de server (zo'n periode wordt in de wachttijdentheorie een busy period genoemd). Herschrijven geeft als verband tussen $E(continu)$ en $E(leegloop)$

$$E(continu) = E(leegloop)\, U/(1-U)$$

Een tijdje nietsdoen begint als de laatste klant uit een periode van continu werken juist vertrokken is en eindigt zodra een volgende klant zich meldt. Klanten komen binnen met onderlinge tussenpozen, die een bepaalde verdeling hebben. De tijdsduur tussen het moment waarop de laatste klant vertrekt en het eerstvolgende moment van aankomst kan gezien worden als een restduur, vanaf het laatste vertrek wordt gewacht hoelang de toestand 'klant komt er aan' duurt.

Vaak zijn de aankomsten volgens een Poissonproces. In het hoofdstuk SIGNALEN EN POISSONPROCES (§12.4) zullen we zien dat dan vanaf elk moment —dus ook vanaf het laatste vertrek— de gemiddelde restduur tot de volgende aankomst gelijk is aan de gemiddelde tussenpoos tussen twee aankomsten. Bij zo'n aankomstpatroon is

$$E(leegloop) = E(tussenpoos)$$

en daarmee

$$E(continu) = E(tussenpoos)\, U/(1-U)$$

Dit is een bekende uitdrukking uit de wachttijdentheorie voor de gemiddelde busy period in de M/G/1 systemen.

8.7.1. voorbeeld: klanten-achter-elkaar

Een DASD verwerkt een doorstroom aan opdrachten van 10 per msek. (DASD is de IBM term voor DIRECT ACCESS STORAGE DEVICE, dus disk). We onderstellen dat de opdrachten binnenkomen volgens een Poissonproces (§12.4). De gemiddelde tussenpoos tussen twee opdrachten is

$$E(tussenpoos) = 1/10 \text{ msek.}$$

Stel dat de DASD 90% van de tijd bezet is. Nu is

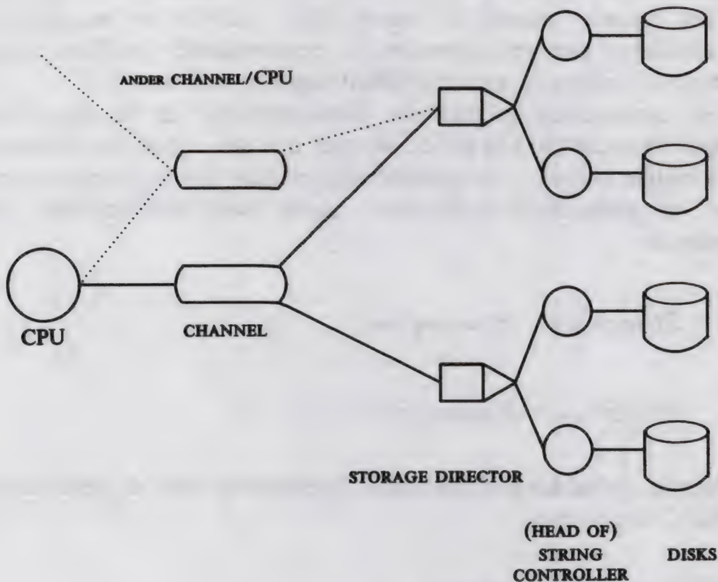
$$E(continu) = 1/10 \times 0.90/0.10 = 9/10 \text{ msek.}$$

(er worden gemiddeld 10 opdrachten achterelkaar verwerkt; §15.8).

8.8. Rotational Position Sensing

8.8.1. missed reconnects

Bij moderne grote mainframes is de opbouw van het subsysteem dat de verbinding verzorgt tussen enerzijds de disks en de snelle andere opslagmedia en anderzijds de CPU en het geheugen, meestal vrij complex. In minicomputers zijn enkele schijven via een diskcontroller verbonden met de configuratie. Zo eenvoudig is het niet bij prijziger systemen.



Figuur 8.3. Channels en controllers.

In grotere 'DASD' systemen zijn een aantal disks verbonden tot een string, over een gemeenschappelijke verbinding worden de disks uit de string bestuurd door een string controller. Deze string controller heet vaak 'head of string'. Een aantal van deze string controllers vormt een groep, de string controllers uit zo'n groep worden bestuurd door en zijn verbonden met één storage director (control unit). Een storage director kan verbindingen hebben met diverse channels (kanalen). Zie figuur 8.3.

Doordat onderdelen uit deze hiërarchische structuur samen gebruikt worden ('geshared'), ontstaan soms erg ingewikkelde 'I/O-configuraties', die moeilijk te analyseren zijn; en die dan ook soms niet goed presteren. We bekijken een bepaald belangrijk aspekt, dat de responsetijd van een DASD sterk kan beïnvloeden. De string controller noemen we kortweg de controller.

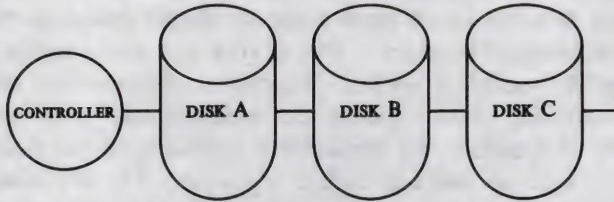
Bij een acces op een disk wordt eerst de arm gepositioneerd (seek), vervolgens wordt gewacht tot de goede sector langs komt (rotational delay) en tenslotte wordt de informatie overgebracht (transfer). Het is voor een goede performance steeds zaak zoveel mogelijk parallel te werken. Daarom is standaard de verbinding met de disk —de controller dus— tijdens de seekfase vrij voor ander verkeer: besturingssignalen of transfers. Bij 'verouderde' uitvoeringen van disks eist de disk de verbinding voor zich op zodra de seek is afgelopen. De verbinding wordt echter pas benut tijdens de transfer. In de rotational delay fase wordt de verbinding dan niet nuttig gebruikt. De parallelliteit kan in principe verder worden opgevoerd door de verbinding pas te claimen als de rotational delay voorbij is. Moderne disks hebben Rotational Position Sensing (RPS) en zijn inderdaad tijdens zowel de seek als de rotational delay los van de verbinding en de controller, pas tijdens de transfer hebben ze de controller continu in gebruik. Op het moment dat de goede sector langs komt moet er bij die disks een pad vanaf de disk naar zijn CPU vrij zijn, anders treedt er een *missed reconnect* op en moet een hele omwenteling worden gewacht voordat de volgende poging kan worden gewaagd. Het aansluiten is nu veel kritischer dan in de vroegere uitvoering, er is maar een kort moment waarop de verbinding kan worden 'gepakt'.

De hele omwentelingstijd aan extra verwerkingsduur die optreedt bij een 'missed reconnect' vormt een ernstige vertragende faktor. En helaas treedt dit juist op als het druk is, want dan zal de verbinding vaak bezet zijn! In de praktijk nemen deze missed reconnects een aanzienlijk deel van de effektieve diskverwerkingsduren voor hun rekening. We willen proberen een redelijke schatting te maken van deze vertraging. Daarbij letten we alleen op de missers waarbij op het moment dat de goede sector langskomt de controller niet vrij is. Een missed reconnect kan in het algemeen ook optreden doordat op dat cruciale moment een channel of een storage director of control unit bezet is, waardoor er ook geen 'pad' vanaf de disk beschikbaar is. Die pechkansen maken de vertraging door missed reconnects groter. Het is een complexe zaak alle bijdragen er aan uit te rekenen.

8.8.2. berekening vertraging

We gaan de gemiddelde vertraging door missed reconnects bij RPS-disks (missed reconnect delay) voor een karakteristiek geval doorrekenen.

Drie disk(unit)s, A, B en C (een 'string'), zijn aangesloten op één (head of string)controller over één verbinding (zie figuur 8.4). Een transfer van A via de controller over de verbinding duurt gemiddeld 20 msek., een transfer van B 30 msek. en een transfer van C 10 msek. De controller verwerkt gemiddeld 2 transfers van A tegen 1 transfer van B en 3 transfers van C. De controller is gedurende 3/4 van de tijd bezig met het verwerken van transfers van A, B of C. Alle drie de disks hebben een omwentelingstijd (rotation time) van 10 msek.



Figuur 8.4. String van (string)controller en drie disks.

De kans dat de controller op een lukraak moment bezig is met een transfer van A is (§7.3)

$$p_A = 3/4 \times \frac{20/3}{20/3 + 30/6 + 10/2} = 0.30$$

De disk A houdt de (string)controller dus 30% van de tijd bezig. Voor B en C is dit 9/40 of 22.5%.

We bekijken de zaak verder vanuit het gezichtspunt van de disk die aan het eind is van zijn rotational delay en ogenblikkelijk de verbinding moet hebben. Omdat er niets precies onderling gesynchroniseerd is, is het redelijk te veronderstellen dat zo'n kritisch moment op een ten opzichte van het verdere gebeuren lukraak tijdstip optreedt (§7.6). We vragen nu naar de kans dat de verbinding bezet wordt aangetroffen.

Dit is niet de kans van $U = 3/4$ dat de verbinding, in casu de controller, bezet is. Want als A om de verbinding vraagt, was hij kennelijk niet bezig met een transfer, het bezet zijn kan alleen van B of C komen, een van die twee kan bezig zijn met een transfer. Het juiste antwoord vinden we het vlotst via operationele schatters. We bepalen de kans dat de verbinding bezet is op het moment dat A om de verbinding vraagt, met de operationele schatter

$$\frac{WERK_B + WERK_C}{WERK_B + WERK_C + IDLE}$$

(notatie als vanouds; $WERK_B$ is de tijd in de *MEETDUUR* dat B de verbinding bezet houdt). Nu delen we overal door *MEETDUUR* en vinden (we stomen direkt door naar de 'echte' waarden)

$$\frac{p_B + p_C}{p_B + p_C + p_{idle}} = \frac{U - p_A}{1 - p_A}$$

(met $U = p_A + p_B + p_C$ als de bezettingsgraad van de verbinding). Deze kans kan

ook worden uitgerekend via de standaardherleidingen voor voorwaardelijke kansen. Hier is de waarde

$$\frac{9/40 + 9/40}{9/40 + 9/40 + 1/4} = 9/14$$

De kans de controller bezet aan te treffen is dus anders dan de bezettingsgraad, omdat wie de verbinding vraagt nooit aanloopt tegen de bezetting die hij zelf veroorzaakt. Er is bij A niet een kans van 3 op 4 maar van 9 op 14 dat er een missed reconnect optreedt.

Na de omwentelingstijd van 10 msek. komt de nieuwe poging. De kans op een 'mis' zal in principe dan anders zijn dan zojuist, omdat er de zekerheid is dat er 10 msek. geleden een transfer van B of C liep (als de transfertijd bijvoorbeeld erg lang is vergeleken met de rotational delay zal de kans op een 'mis' vrij groot zijn, groter dan 9 op 14). Als we even gemakshalve veronderstellen dat er echter ook bij de nieuwe poging en bij elke volgende een kans is van 9 op 14 op 'mis', duurt het gemiddeld 14/5 pogingen voordat met de transfer van A kan worden begonnen. De kans op succes is immers telkens maar 5 op 14, en dan zegt de van de binomiale verdeling afgeleide geometrische verdeling (en het gezond verstand) dat het gemiddeld 14/5 beurten duurt voordat er succes optreedt. De aanname van een constante kans op een missed reconnect bij elke volgende poging blijkt in de praktijk heel redelijk te werken.

Doordat ook disk B en disk C gebruik maken van dezelfde (string)controller treedt er dus in de verwerkingsduur bij disk A een extra bijdrage op van $(14/5-1)\times 10$ msek. = 18 msek. (de -1 omdat de laatste poging raak is).

Bij een bezettingsgraad van de 'geshared' controller van 3/4 of 75% zal de toename in de verwerkingsduur door missed reconnects voor disk A dus 18 msek. zijn bij de huidige werklast. Voor disk B en disk C is de toename volgens dezelfde berekening 21 msek.

8.8.3. voorbeeld: maximale bezettingsgraad van een controller

Vaak wordt door systeembeheerders als vuistregel aangehouden dat de totale rotational delay door de missed reconnects niet meer dan verdubbeld mag worden; de gemiddelde extra verwerkingsduur zou hoogstens een halve omwentelingstijd mogen zijn. Welke bezettingsgraad (U) mag de controller dan hoogstens hebben?

Het verkeer over de drie disks wordt gewoonlijk door beheerders gebalanceerd gehouden: iedere disk levert dezelfde bezettingsgraad: $p_A = p_B = p_C = U/3$. Laten we dit veronderstellen.

Volgens het voorgaande zal een disk, die de verbinding probeert te pakken, de controller dan bezet vinden in het deel

$$Q = \frac{U - 1/3\,U}{1 - 1/3\,U}$$

van de gevallen. De gemiddelde extra bijdrage is $(1/(1-Q)-1) = Q/(1-Q)$ maal de omwentelingstijd; deze zou hoogstens de halve omwentelingstijd mogen zijn. Daarom moet

$$Q/(1-Q) \leq 1/2$$

of $Q \leq 1/3$ en $U \leq 3/7$. De bezettingsgraad van een controller zal dus betrekkelijk gering moeten zijn.

8.9. GEMIDDELDE RESPONSETIJD DISK

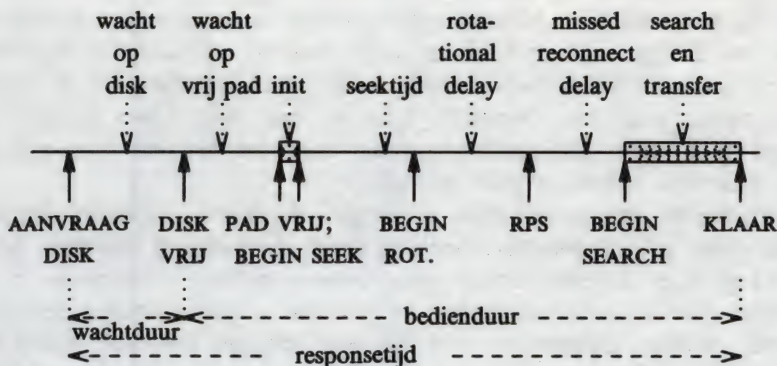
Als er in een programma een I/O-opdracht voor een bepaalde disk wordt gegeven, duurt het geruime tijd voordat deze opdracht is afgewerkt, ten minste wanneer de te behandelen data niet al in de I/O-cache in het geheugen staan.

De tijd tussen het genereren van de I/O-opdracht en het verwerkt zijn er van is in termen van wachttijdensystemen (zie §14.7.2) de verblijfduur bij de 'diskserver'. Het tijdinterval wordt in dit geval ook wel —kortweg, maar wat verwarrend— de responsetijd voor de I/O-opdracht genoemd. We willen aangeven hoe het gemiddelde van deze responsetijd kan worden berekend.

Er zijn heel wat faktoren en duren die de responsetijd beïnvloeden, maar ze hebben merendeels een geringe invloed in een goed getuned systeem. Er zijn eigenlijk maar enkele grotere bijdragen. Maar het is jammer genoeg soms nogal lastig deze goed te voorspellen, vooral als de meetgegevens onvolledig zijn.

We bekijken de levensloop van een I/O-opdracht vanaf zijn ontstaan bij het programma (zie figuur 8.5). De I/O-opdracht probeert te bereiken dat er informatie, die op één van de honderden disks staat, in het geheugen wordt gezet (of omgekeerd). Er kunnen op het moment van ontstaan van de opdracht al andere opdrachten voor dezelfde disk uitstaan. Een van die opdrachten wordt uitgevoerd, de anderen wachten. Onze opdracht zal nu eerst moeten wachten totdat al deze opdrachten zijn afgewerkt. Dit is wachten in het wachttijdensysteem 'diskserver', we noemden dit de *'wachtduur'*.

Zodra de voorgangers allemaal zijn afgewerkt komt de beurt aan onze I/O-opdracht. Er gaat nu een initialisatiesignaal naar de disk; met het signaal wordt aangegeven wat er moet gebeuren, zodat akties kunnen worden geprepareerd. Het signaal zal over een van de paden channel-storage director-(string)controller-disk moeten lopen. Dit signaal zal daarom soms moeten wachten tot er een pad naar de disk vrij is, de vertraging is de *'signaalwachtduur'*. Als het signaal is aangekomen volgt de bekende sequentie *seektijd* plus *rotational delay*. Dan is het moment voor de reconnect aangebroken. Als er op dat moment geen pad vrij is van de disk naar de CPU (disk-(string)controller-storage director-channel-CPU) mislukt de reconnect. Er moet dan één of meer omwentelingen worden gewacht *(missed reconnect delay)*. Bij een vrij pad volgt de *search*-fase, waarin de plaats van de data op de disk getraceerd worden. Tenslotte wordt de informatie langs het



Figuur 8.5. Bestanddelen responsetijd disk (gearceerd: verbinding fysiek bezet; vette pijlen: wijziging toestand). Eerst wordt gewacht totdat alle voorgaande opdrachten voor de disk verwerkt zijn (wacht op disk; wachtduur bij diskserver). Hierop wordt een initialisatiesignaal naar de disk gestuurd, dit moet wachten tot er een pad over channel, storage director en controller vrij komt (wacht op vrij pad; signaalwachtduur). Door het initialisatiesignaal wordt de gewenste positie op de disk gespecificeerd (init, omvat controller protocol). Na de seektijd en de rotational delay wordt op het moment RPS de reconnect gemist, zodat pas na de missed reconnect delay aan de search kan worden begonnen. Tenslotte vindt de transfer van de data plaats.

pad van disk naar geheugen (of omgekeerd) overgehaald: de *transfer*.

Tussen het begin van het initialisatiesignaal —het moment waarop de opdracht bij de diskserver in behandeling wordt genomen— en het overgebracht zijn van de informatie —gereedmelding van de diskserver— verloopt een tijd, die we de bedienduur van de diskserver noemen. Deze bedienduur zal bestaan uit signaalwachtduur, seektijd, rotational delay, missed reconnect delay, search- en transfertijd, plus de extra tijd dat de controllers (de tussenstations) bezig zijn met het doorgeven van signalen en het prepareren van akties. Merk op dat twee van deze bijdragen komen door 'pad bezet': op de weg heen als het initialisatiesignaal moet wachten en op de weg terug bij de missed reconnects. De laatste bijdrage zal een veel grote vertraging betekenen omdat het kunnen profiteren van het pad zo kritisch ligt. We bekijken alle tijden afzonderlijk.

Bij disks uit grote mainframes wordt niet geprobeerd de seektijd en de rotational delay in te korten door optimale schedulingsalgoritmen. Er is niet zoiets als SHORTEST SEEK TIME FIRST bij seeks. En de data worden ook niet 'rotationeel handig' op de cylinders geplaatst om er voor te zorgen dat er bij sequentieel zoeken op dezelfde cylinder vrijwel geen rotational delay ontstaat. De gemiddelde seektijd en rotational delay zijn dan ook in principe eenvoudig aan te geven, gegeven het patroon van aanvragen.

Over de gemiddelde seektijd worden door de fabrikant van de disk gegevens verstrekt. De door hem opgegeven gemiddelde seektijd is de gemiddelde seektijd

als de plaatsen, waar de kop moet zijn, bij opeenvolgende diskopdrachten lukraak over de disk verdeeld zijn. Meestal is het patroon dat zich voordoet heel anders. Er zijn betrekkelijk veel opdrachten voor dezelfde cylinder, bijvoorbeeld bij sequentieel acces. Zulke I/O-opdrachten kosten geen seektijd. Of er wordt veel teruggesprongen naar een bepaalde cylinder om daar bijvoorbeeld 'catalog' informatie op te halen. Om een betrouwbaar beeld van het seek-verkeer te krijgen zal echt de monitor moeten worden gebruikt!

De gemiddelde rotational delay is simpel de halve omwentelingstijd. De rotational delay is een restduur: hoelang duurt de toestand 'gezochte info niet onder de kop' nog? De schijf draait met constante snelheid. Het eind van de seek is vergeleken met het moment, waarop de gezochte informatie onder de kop komt, een lukraak moment. De restduur heeft dus een uniforme verdeling tussen 0 en de omwentelingstijd. Kortom, het is het eenvoudigste geval van de formule voor de gemiddelde restduur.

De gemiddelde search- en transfertijden hangen af van verdere specificaties, zoals de blockgrootte. Met de door de fabrikant verstrekte gegevens zijn ze dan uit te rekenen.

Van alle tijd waarin de controllers voor de opdracht bezig zijn, is voor de responsetijd alleen de tijd relevant die niet 'overlapped' wordt door andere componenten uit de bedienduur. Anders worden er tijdsduren dubbel geteld. De fabrikant geeft hier weer details en precieze informatie. We noemen deze vertraging de duur van het *'controller protocol'*.

De signaaldruren zijn alle erg kort. In moderne configuraties is er weinig vertraging doordat signalen moeten wachten. In §8.5.2 bepaalden we de relevante gemiddelde 'signaalwachtduur' voor een karakteristiek geval.

Een reconnect kan niet tot stand komen als er geen pad vrij is van disk naar CPU. Paden worden bezet door I/O-verkeer voor andere disks voor mogelijk andere CPU's; ze worden gekozen volgens een algoritme van de fabrikant. De bezettingsgraden van de paden hangen sterk af van de drukte in de hele I/O-configuratie. Voor het bepalen van de vertraging door missed reconnects is het nodig uit te rekenen hoe groot de kans is dat alle paden bezet zijn. Dat is geen eenvoudig probleem, we gaan er niet verder op in. In §8.8.2 bepaalden we een onderdeel er van.

De voortschrijdende technologie maakt dat I/O-opdrachten elkaar gelukkig steeds beter kunnen ontlopen. Bij MVS-XA van IBM bijvoorbeeld hoeft het pad terug niet gelijk te zijn aan het pad heen, zoals dat vòòr het uitbrengen van MVS-XA wel het geval was (het pad van transfer moest ongelukkigerwijs hetzelfde zijn als het pad van het initialisatiesignaal). De introduktie van controllers volgens bijvoorbeeld 'Device Level Selection' vergroot het aantal paden aanzienlijk. Zo blijft de kans op een 'missed reconnect' bij een goede tuning ook in de moderne complexe I/O-configuraties binnen de perken. Een waarde van 0.3 zal steeds haalbaar zijn.

Als concreet voorbeeld nemen we de gemiddelde responsetijd voor een 3380 disk van IBM. Deze disk heeft volgens de specificaties een gemiddelde seektijd van 16.0

msek. en een omwentelingstijd van 16.6 msek.; search+transfer duurt (bij blockgrootte 4KB) ongeveer 1.5 msek. De gemiddelde duur van het 'controller protocol' is 1.0 msek.

We geven de gemiddelde responsetijd onder omstandigheden, waarin de kans op een missed reconnect 1/3 is. De extra delay is dan $16.6 \times (1/3)/(2/3) = 8.3$ (§8.8). Voor de 'signaalwachtduur' nemen we de voor het voorbeeld uit §8.5.2 bepaalde waarde van 0.44 msek., deze is zo klein dat we hem eigenlijk weer prompt moeten verwaarlozen. Als gemiddelde seektijd nemen we 16.0/4 msek.: er zijn nogal wat seeks naar dezelfde cylinder.

De gemiddelde bedienduur voor een I/O-opdracht is hiermee (seektijd + rotational delay + missed reconnect delay + search en transfer + controller protocol + signaalwachtduur)

$$gem.\ bedienduur = 4.0 + 8.3 + 8.3 + 1.5 + 1.0 + 0.4 = 23.5 \text{ msek.}$$

De gemiddelde responsetijd bestaat uit deze tijd, plus de tijd die aanvankelijk bij de server 'diskserver' wachtend wordt doorgebracht, doordat de disk nog bezig is met eerdere diskopdrachten. Deze gemiddelde wachtduur moet nog worden aangegeven.

In de praktijk blijkt een modellering van het wachttijdensysteem 'diskserver' als M/M/1 systeem redelijk te voldoen. De zojuist berekende gemiddelde bedienduur komt dan overeen met de gemiddelde serviceduur uit het M/M/1 systeem. De uitdrukking voor de gemiddelde verblijfduur in zo'n systeem staat in §8.6. De gemiddelde responsetijd (dat is de verblijfduur bij de diskserver) volgt daarmee uit de zojuist berekende duur van 23.5 msek. door te vermenigvuldigen ('op te blazen') met de factor $1/(1-U)$.

Als de disk 30% van de tijd wordt gebruikt is de gemiddelde responsetijd

$$gem.\ responsetijd = 23.5/0.7 = 33.6 \text{ msek.}$$

8.10. SAMENVATTING

In dit hoofdstuk hebben we gezien dat de gemiddelde restduur in allerlei vermommingen op veel plaatsen opduikt. Het is nuttig dat er zo'n eenvoudige vlotte inzichtelijke formule voor is. De restduur kan in principe altijd worden uitgerekend door de basisrelatie voor de kansen op optreden en aantreffen uit het hoofdstuk BEURTEN EN KANSEN toe te passen. Dit levert één van de vormen (8.1), (8.2) en (8.3) voor de gemiddelde restduur op; anders is de indirekte vorm (8.4) te gebruiken.

De bekendste uitdrukking is

$$E(restduur) = \frac{E(duur^2)}{2E(duur)}$$

De vertraging die bij wachten wordt opgelopen wordt sterk bepaald door de gemiddelde restduur totdat de server voor het eerst na aankomst vrij komt. In veel gevallen is deze wachtrestduur de gehele wachtduur. In andere gevallen is de totale wachtduur vaak te vinden door de gemiddelde wachtrestduur te vermenigvuldigen met de factor $1/(1-U)$.

In moderne I/O-configuraties treedt op veel plaatsen een extra vertraging op door wachten. Deze bijdragen aan de responsetijd kunnen worden afgeschat.

8.11. OPGAVEN

opgave 8.1: vervolg RPS (8.8.2)

Neem aan dat de transfertijden van *B* en *C* precies 30 en 10 msek. zijn.

- Disk *A* vindt bij zijn poging de verbinding bezet. Hoelang duurt het gemiddeld voordat de 'lopende' transfer is afgelopen (10 msek.)?
- En hoelang als de transfertijden negatief exponentieel verdeeld zijn (20 msek.)?
- Een verzoek van buiten merkt dat het gemiddeld *x* msek. duurt voordat de lopende transfer voorbij is. Hoe groot is *x*? Wat is de gemiddelde transfertijd? Hoe groot is de gemiddeld door het verzoek aangetroffen transfertijd (20 msek.)? Wat moet worden aangenomen?

opgave 8.2: vervolg interrupt-monitoring (opgave 7.2, 12.5.5)

Er zijn 3 soorten interrupts. Van alle gegenereerde interrupts is 1/6 deel van soort *I*, de verwerkingsduur van deze interrupts (duur van de routine) is gemiddeld 30 msek., negatief exponentieel verdeeld. Verder is 1/3 deel van soort *II* met een duur van gemiddeld 15 msek., uniform verdeeld tussen 13 en 17 msek. en de helft van soort *III* met een duur van exact 20 msek. Men wil graag weten met welke interrupt de CPU bezig is en zendt daarom een hardware-signaal, dat op een lukraak moment constateert dat de CPU bezig is.

- Hoelang duurt het gemiddeld nog voordat de aangetroffen interrupt is afgewerkt?

opgave 8.3: multiplex verbinding

Over een zekere verbinding loopt veel berichtenverkeer afkomstig van diverse locaties, waaronder de locatie Honhio. De verbinding wordt door de locaties 'in de tijd gedeeld' (gemultiplext). Verkeer dat zich op een bepaalde plaats aanbiedt moet wachten tot de verbinding beschikbaar komt. Zodra dit het geval is worden alle lokaal verzamelde berichten achter elkaar over de verbinding getransporteerd. Het transport van een bericht duurt gemiddeld 1 msek.

Nadat alle verzamelde berichten verstuurd zijn, duurt het enige tijd tot de verbinding weer ter beschikking komt van Honhio. Zo'n tussenpoos duurt in 1 van de 4 gevallen precies 2 sekonden en in 3 van de 4 gevallen is de tussenpoos negatief exponentieel verdeeld rond een gemiddelde waarde van 4 sekonden.

Men vraagt zich af hoelang het duurt voordat een bij de locatie Honhio binnengekomen bericht is overgestuurd. We noemen dit de verblijfduur van het bericht. Er komt gemiddeld 1 bericht per 3.5 sekonden bij Honhio binnen, berichten komen op lukrake momenten.

- Bepaal (tot op enkele msek.) de gemiddelde verblijfduur van het bericht (3 4/7).
- Hoeveel berichten wachten er gemiddeld bij Honhio op transport (1 1/49)?

opgave 8.4: paradox: busy period in M/M/1

In een M/M/1 systeem is de gemiddelde tijd die een binnenkomende klant in het systeem doorbrengt, gelijk aan de gemiddelde duur van een periode van ononderbroken (continu) werken van de server (de busy period; vergelijk §8.7, §15.8-9). Iemand merkt op dat dit een paradoxale uitkomst is, omdat ofwel een binnenkomende klant op een moment komt waarop de server zonder werk is en hij dus direkt geholpen wordt, ofwel binnenkomt tijdens een busy period en dan niet de hele busy period in het systeem zal zitten. Daaruit wil hij besluiten dat de gemiddelde verblijfduur van een binnenkomende klant korter moet zijn dan de gemiddelde duur van een busy period.

Dien hem van repliek.

9

Ergodische Markovketens

9.1. INLEIDING

In de loop van de tijd blijft er weinig hetzelfde. Veranderingen komen nogal eens plotseling en met schokken, maar soms lijkt het alsof iets heel geleidelijk anders wordt. De tijd kan —voor de waarnemer— stapsgewijs of continu verstrijken. Stapsgewijze veranderingen brengen een systeem van de ene discrete toestand in de andere, continue veranderingen wijzigen de toestand van een systeem voortdurend. Hoe men de veranderende tijd ziet is vooral een filosofisch probleem.

In dit hoofdstuk sluiten we ons aan bij hen die alleen stapsgewijze veranderingen willen zien. Een optredende continue wijziging wordt beschouwd als een later aan te brengen correctie op het in eerste instantie 'schoksgewijs' variërende beeld. Bij het analyseren van wat zich in en om een computerconfiguratie afspeelt ligt dit gezichtspunt voor de hand. Ook in de voorgaande hoofdstukken was het ons uitgangspunt. Redeneringen die uitgaan van een continu veranderingsproces komen in het hoofdstuk SIGNALEN EN POISSONPROCES aan de orde.

Vaak is het mogelijk in elke voorkomende toestand aan te geven wat de volgende toestanden kunnen zijn en hoe vaak deze gemiddeld als volgende toestand optreden. In de theorie van de Markovketens wordt aangegeven hoe die informatie kan worden benut om het globale gedrag in de tijd te beschrijven. In het hoofdstuk BEURTEN EN KANSEN interesseerde het ons hoe vaak de toestanden aan bod kwamen, nu gaan we ook de afwisseling van de toestanden na.

Het blijkt dat Markovketens berekeningen mogelijk maken voor de *kansen op optreden*, de kansen *op de beurt*, die we algemener —maar wat vaag— in BEURTEN EN KANSEN beschreven.

De aanpak volgt eerst volledig de stochastische weg vanuit voorwaardelijke kansen op toestanden. Naderhand —in het hoofdstuk MARKOVKETENS EN

EVENWICHTSVERGELIJKINGEN— wordt de operationele zienswijze gepresenteerd, die ook hier zijn nut blijkt te hebben.

In de computer-prestatieanalyse speelt het begrip *stationaire toestand* of *evenwichtstoestand* een belangrijke rol. Het blijkt in dit hoofdstuk dat bij ergodische Markovketens nauwkeurig kan worden aangegeven wat er mee wordt bedoeld. In de stationaire fase voldoen de kansen op optreden van de diverse toestanden —de kansen op de beurt— aan de evenwichtsvergelijking

$$\pi = \pi \mathbf{P}$$

In de overgangsmatrix **P** staan de kansen om van de ene toestand naar de andere te gaan. De kansen op optreden in de stationaire toestand zijn samengevoegd tot de stationaire toestandsvector π. De elementen uit π zijn de kansen *op de lange termijn*, het blijken ook de kansen *over de lange termijn* te zijn; ze geven aan hoe vaak de diverse toestanden relatief bezet zijn.

In dit hoofdstuk behandelen we de stationaire toestand van ergodische Markovketens. We beginnen met een standaardvoorbeeld, waaraan we de beschouwingen, afspraken en inzichten demonstreren. Het simpele voorbeeld had kunnen worden ontleend aan de informatica —we brengen het niet zo, maar refereren aan een bezigheid waarin het nog waarschijnlijker is dat absurde situaties ook metterdaad telkens weer worden gerealiseerd.

9.2. STANDAARDVOORBEELD: SCHUTTERS

9.2.1. schutters

We beschouwen een proces uit de semi-militair-folkloristische praktijk. Bij een grootscheepse oefening zijn 1000 schutters aan het schieten. We beginnen op het moment dat alle 1000 zojuist raak hebben geschoten† en proberen het verdere verloop van de oefening te voorspellen.

Hiertoe zullen de nodige gegevens beschikbaar moeten zijn. Als de volledige beschrijving en de werking van het schiettuig bekend is, de manier waarop het richten wordt ingesteld, de snelheid van de schutter en de snelheid van het doel, enzovoort, enzovoort, is het mogelijk om met behulp van de wetten van de natuurkunde precies uit te rekenen wat er bij de volgende schoten gaat gebeuren.

De informatie is echter zelden zo volledig beschikbaar. Men zal zich moeten behelpen met onvolledige informatie. Zo weet een geroutineerde schutter —we nemen een voorbeeld— dat hij na raak te hebben geschoten, opnieuw raak schiet in '80 van de 100' gevallen. Net zo weet hij dat een misser gemiddeld even vaak gevolgd wordt door een raak schot als door weer een misser. Voor een

† Aannemend dat zoiets zich inderdaad kan voordoen.

geroutineerde schutter is dus de kans om na een raak schot opnieuw raak te schieten gelijk aan 4/5 en de kans om na een misser raak te schieten is 1/2.

Deze uit ervaring opgedane onvolledige kennis nemen wij over in een model van de manier, waarop deze schutters funktioneren. Een schutter kan raak schieten of missen. We zullen zeggen dat hij zich na een raak schot in de toestand *raak* (genummerd met 0) bevindt en na een gemist schot in de toestand *mis* (genummerd met 1). Bij het volgende schot kan de schutter opnieuw raak schieten of missen, hij zal zich opnieuw of wel in de toestand 'raak' dan wel in de toestand 'mis' bevinden. Het verloop van de oefening kan voor een bepaalde schutter bijvoorbeeld als volgt zijn:

(eerste 16 schoten) 0,0,1,0,1,0,0,0,1,0,0,0,0,1,0,0

Deze schutter schiet onder andere bij het 3-de, 5-de, 9-de en 14-de schot mis. Een andere schutter zal heel anders presteren. We proberen het verloop van de oefening aan te geven voor de 'gemiddelde schutter' (maar deze uitdrukking uit het gewone spraakgebruik is voor onze beschouwingen helaas lang niet goed genoeg omschreven).

We zijn alleen geïnteresseerd in en hebben kennis over het raak of mis schieten van een schutter. Vandaar dat we van een schutter alleen opgeven of hij raakt of mist. Meer niet.

Als een ervaren schutter eerst raak schiet en daarna mist gaat hij over van de toestand 'raak' in de toestand 'mis'. Schiet hij opnieuw raak, dan gaat hij over van de toestand 'raak' in de toestand 'raak'. De kans dat dit laatste gebeurt zou gelijk zijn aan 4/5, de kans dat het eerste gebeurt is dan gelijk aan $1-4/5= 1/5$ (som van de kansen is 1). Net zo zouden zowel de kans op een overgang van 'mis' naar 'mis' als van 'mis' naar 'raak' beide gelijk zijn aan 1/2.

Deze gegevens worden op een compacte manier weergegeven door het schema in figuur 9.1.

	raak	*mis*
raak	4/5	1/5
mis	1/2	1/2

Figuur 9.1. Schema overgangen schutters.

Als we het getallenblok uit figuur 9.1 als een matrix opvatten, noemen we het de *overgangsmatrix*. De overgangsmatrix beschrijft in het kort wat er in het model wordt opgenomen over het funktioneren van een ervaren schutter.

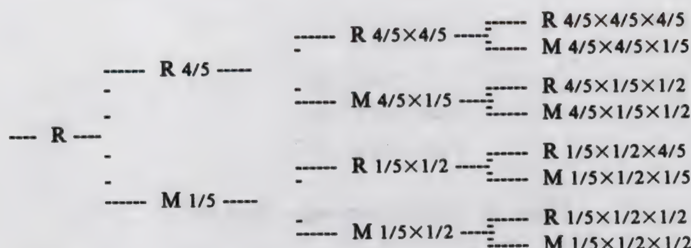
9.2.2. voorspellingen

Wat zijn de voorspellingen van dit model over het verloop van de oefening?

Alle 1000 'ervaren' schutters bevinden zich op het moment dat we gaan kijken in de toestand 'raak'. Iedere schutter heeft een kans van 4/5 om bij het volgende schot opnieuw raak te schieten. Is het dan zo dat van de 1000 schutters er 800 raak en 200 mis schieten? Nee, we moeten het woordje 'gemiddeld' toevoegen. Er zullen gemiddeld 800 raak en gemiddeld 200 mis schieten als er heel vaak groepen van 1000 schutters schieten met een kans van 4/5 op succes. Als onze 1000 schutters schieten kunnen er best maar 775, maar misschien zelfs 825 raak schieten. Meestal zullen er echter wel ongeveer 800 raak schieten —volgens de wet van de grote aantallen. Gemakshalve zegt men toch wel eens dat er in zo'n geval 800 schutters raak zullen schieten en laat men het woord gemiddeld dus weg, volgens 'verkort' spraakgebruik.

In het begin (nulde schot, $n=0$) hebben alle schutters raak geschoten. Bij het eerstvolgende schot (eerste schot, $n=1$) zullen gemiddeld 800 raak en 200 mis schieten. Van de gemiddeld 800 die bij het eerste schot raak schieten zullen er een aantal vervolgens opnieuw raken. Er zullen gemiddeld 640 ($= 4/5 \times 4/5 \times 1000$) na een raak eerste schot opnieuw raak schieten en gemiddeld 160 ($= 1/5 \times 4/5 \times 1000$) zullen missen (tweede schot, $n=2$). Van de gemiddeld 200 die zullen missen bij het eerste schot schieten er gemiddeld 100 ($= 1/2 \times 1/5 \times 1000$) bij het tweede schot raak en gemiddeld evenveel missen er. In totaal zal gemiddeld door 740 ($= (4/5 \times 4/5 + 1/2 \times 1/5) \times 1000$) schutters bij het tweede schot raak worden geschoten en gemiddeld 260 ($= (1/5 \times 4/5 + 1/2 \times 1/5) \times 1000$) schutters zullen dan missen.

Op dezelfde manier berekenen we de situatie na het derde schot ($n=3$) en voor algemene n. Figuur 9.2 geeft een overzicht van de berekening in de vorm van een 'boom'. Bij elk volgend schot ontstaan telkens twee nieuwe mogelijkheden, vandaar dat elke 'tak' zich steeds in tweeën splitst.



Figuur 9.2. Voorspelling voor het verloop van de oefening tot en met het derde schot; er staan kansen op raak (R) en mis (M).

Zo'n berekening wordt snel erg uitgebreid. Laten we ons beperken tot het gemiddeld aantal schutters dat bij een bepaald schot raak schiet, dan wel mist. Hoe vinden we bijvoorbeeld het gemiddeld aantal schutters dat raak schiet bij het

derde schot? We gaan uit van de situatie na het tweede schot, waarbij er gemiddeld 740 schutters raak schieten en 260 schutters missen. Van de schutters die raken, schiet bij het 3-de schot gemiddeld het 4/5 deel raak, zodat er gemiddeld 592 (= 4/5×740) bij het 2-de en het 3-de schot raken en er gemiddeld 148 (= 1/5×740) bij het 2-de schot raken en bij het 3-de missen. Er zijn gemiddeld 130 schutters die missen bij het 2-de schot en raken bij het 3-de en gemiddeld evenveel missen bij beide schoten. In totaal schieten er bij het 3-de schot gemiddeld 722 (= 592+130) raak en 278 (= 148+130) mis. Uitgaande van deze getallen kunnen we berekenen hoeveel er gemiddeld bij het 4-de schot raak en mis schieten. Enzovoort.

Zulke berekeningen worden beknopt genoteerd door gebruik te maken van de matrixrekening. Laten we alle optredende aantallen delen door het totaal aantal schutters, dus 1000. De voorspelling van de situatie na het 3-de schot is qua rekenwerk niets anders dan het resultaat van een matrixvermenigvuldiging volgens:

$$(0.722,\ 0.278) = (0.740,\ 0.260)\begin{bmatrix} 4/5 & 1/5 \\ 1/2 & 1/2 \end{bmatrix}$$

De hier optredende matrix is de overgangsmatrix uit figuur 9.1, die we **P** zullen noemen. De getallen 0.740 en 0.260 beschrijven de voorspelde situatie na het 2-de schot —voor wat raak en mis betreft—, de getallen 0.722 en 0.278 de voorspelde situatie na het 3-de schot. Men zegt dat zulke getallen samen een *toestandsvector* vormen. De voorspelde situatie na het eerste schot staat in de toestandsvector (0.800, 0.200). Iedere schutter heeft aan het begin bij zijn voorgaand schot raak geschoten; de begintoestandsvector is (1, 0). De verdere berekening van de toestandsvectoren staat in figuur 9.3.

Wat betekent zo'n toestandsvector? Let maar op de voorspelde situatie na het eerste schot. Ieder van de 1000 schutters heeft een kans van 4/5 om raak te schieten (in toestand 0 te geraken). De getallen (0.800, 0.200) in de toestandsvector na het eerste schot geven niets anders aan dan de kansen voor een schutter, die zojuist —bij het nulde schot— raak heeft geschoten, om in de toestand 'raak' respectievelijk 'mis' terecht te komen na het eerste schot. Net zo geven de getallen uit (0.722, 0.278) de kans aan dat een schutter die zojuist geraakt heeft, bij het derde schot daarna raak schiet, respectievelijk mist. We hebben eerst op 1000 schutters gelet en daarom in het rekenwerk gebruikt dat het gemiddeld aantal schutters, dat in een bepaalde situatie raak zal schieten, wordt gevonden door het aantal schutters te vermenigvuldigen met de kans op raak voor een bepaalde sequentie. Door te delen door het aantal schutters zijn we —gelukkig— weer op kansen terecht gekomen; die kansen staan in de toestandsvector.

Een schutter die bij het 0-de schot raak schiet heeft algemeen een kans $p_n(0)$ om bij het n-de schot raak te schieten en een kans $p_n(1)$ om bij het n-de schot te missen. De vector $p_n = (p_n(0), p_n(1))$ is de toestandsvector na het n-de schot (na de

beginsituatie	:	$(p_0(0), p_0(1))$	$= (1, 0)$
eerste schot	:	$(p_1(0), p_1(1))$	$= (0.800, 0.200)$ $= (1, 0)\mathbf{P}$
tweede schot	:	$(p_2(0), p_2(1))$	$= (0.740, 0.260)$ $= (0.800, 0.200)\mathbf{P}$ $= (1, 0)\mathbf{P}^2$
derde schot	:	$(p_3(0), p_3(1))$	$= (0.722, 0.278)$ $= (p_2(0), p_2(1))\mathbf{P}$ $= (p_1(0), p_1(1))\mathbf{P}^2$ $= (p_0(0), p_0(1))\mathbf{P}^3$
n-de schot	:	$(p_n(0), p_n(1))$	$= (p_0(0), p_0(1))\mathbf{P}^n$ $= (p_{n-k}(0), p_{n-k}(1))\mathbf{P}^k$
	:	p_n	$= p_0\mathbf{P}^n$ $= p_{n-k}\mathbf{P}^k$

Figuur 9.3. Berekening toestandsvectoren.

n-de stap). De kansen $p_n(0)$ en $p_n(1)$ worden bepaald door matrixvermenigvuldiging volgens

$$(p_n(0), p_n(1)) = (p_0(0), p_0(1))\mathbf{P}^n$$

met $(p_0(0), p_0(1)) = (1, 0)$: het nulde schot was raak.

De specificatie van de begintoestand kan niet worden weggelaten. We doen voorspellingen uitgaande van een bepaalde beginsituatie (beginvoorwaarde). In de statistiek spreekt men in zulke gevallen van voorwaardelijke kansen, hier gaat het om kansen onder de voorwaarde dat bij het 0-de schot raak is geschoten.

9.2.3. het werken met toestandsvectoren

De toestandsvector na het m-de schot vonden we als

$$(p_m(0), p_m(1)) = (1,0)\mathbf{P}^m$$

De toestandsvector na het (m + k)-de schot is

$$(p_{m+k}(0), p_{m+k}(1)) = (1,0)\mathbf{P}^{m+k}$$

Deze toestandsvector kunnen we ook als volgt vinden, we noemen hem dan $(r_k(0), r_k(1))$

$$(r_k(0), r_k(1)) = (p_{m+k}(0), p_{m+k}(1)) = (1,0)\mathbf{P}^m\mathbf{P}^k$$

dus:

$$(r_k(0), r_k(1)) = (p_m(0), p_m(1))\mathbf{P}^k$$

We kunnen hiermee vragen beantwoorden als: bij een bepaald schot is de kans op raak 2/3 en de kans op mis 1/3; hoe groot is de kans op raak respectievelijk mis bij het 2-de daarop volgende schot? Welnu, neem $k=2$ en $(p_m(0), p_m(1)) = (2/3, 1/3)$, de waarde van m doet er niet toe! De gevraagde kansen worden dan gegeven door

$$(r_2(0), r_2(1)) = (2/3, 1/3)\mathbf{P}^2 = (0.71, 0.29)$$

Dit antwoord zouden we ook gevonden hebben als we niet van de begintoestand (1,0), maar van een andere begintoestand waren uitgegaan. Verandering van de begintoestand verandert niets aan de relatie $(r_k(0), r_k(1)) = (p_m(0), p_m(1))\ \mathbf{P}^k$. Het schot met kansen (2/3, 1/3) vormt een nieuw beginpunt van waaruit de kansen op komende prestaties berekend kunnen worden. Wat er vóór dit schot gebeurde is niet van belang, we hoeven alleen te weten hoe de kansen bij dit schot liggen. Dat vormt een nieuwe beginsituatie, van waaruit verder wordt gerekend.

De kansen op toekomstige situaties worden vanuit de beginsituatie bepaald via een keten van overgangsmatrices. Zo'n keten is —zo zegt men— een *Markovketen*. De tijd wordt bij de schutters geteld in schoten, bij Markovketens spreekt men algemeen van *stappen*; na het n-de schot is na de n-de stap.

Het is dus ook mogelijk aan te geven hoe de voorspellingen in ons systeem zullen liggen als er niet bekend is dat het laatste schot, voordat er voorspellingen worden gedaan, raak was. Wanneer bekend is dat de kansen op raak en mis bij dit schot gelijk zijn geweest aan $(p_0(0), p_0(1))$, moet onze voorspelling voor het n-de schot zijn

$$(r_n(0), r_n(1)) = (p_0(0), p_0(1))\ \mathbf{P}^n$$

Ook 'samengestelde' kansen kunnen op een analoge manier worden berekend. Zo zijn de kansen om bij het 3-de en 5-de schot te raken en bij het 8-ste schot te raken (t(0)) dan wel te missen (t(1))

$$(t(0), t(1)) = (p_0(0), p_0(1))\mathbf{P}^3\,\mathbf{R}\mathbf{P}^2\,\mathbf{R}\mathbf{P}^3$$

De hierin optredende matrix **R** heeft de vorm

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Het is de *projectieoperator* op de toestandsvector raak.

Wanneer er niet 2, maar *n* toestanden mogelijk zijn, is de overgangsmatrix een n-bij-n matrix. De projectieoperator op de i-de toestand is dan de n-bij-n matrix met overal een nul, behalve op de positie i-de rij, i-de kolom, daar staat een 1. De projectieoperator is dus het matrixprodukt van de kolomvector met een 1 op positie *i* en verder overal een 0, en dezelfde vector als rijvector.

9.3. Markovketens

Ons voorbeeld was eenvoudig, we onderscheidden maar twee toestanden. Toch is de algemene situatie niet wezenlijk moeilijker; er zullen alleen veel meer toestanden —en daarin enkele soorten (klassen)— moeten worden onderscheiden.

De aanpak is die van *Markovketens.* Systemen doorlopen *stapsgewijs* wisselende toestanden. Na elke stap kan de toestand gewijzigd zijn. Er heeft dan een echte overgang plaats gehad naar een andere toestand —die is de heersende geworden—, maar de toestand kan ook dezelfde zijn gebleven, de overgang is dan naar dezelfde toestand. Een beschrijving via het begrip beurt is beknopt en bondig: een andere toestand kan de beurt hebben gekregen, of de oude toestand kan na de stap opnieuw de beurt hebben.

Welke toestand de beurt krijgt na een bepaalde stap zou bij volledige informatie over het systeem berekend kunnen worden uit informatie over de voorgeschiedenis. De kennis over het verleden kan hier hoogstens bestaan uit de wetenschap welke toestanden in welke volgorde tot nu toe de beurt hebben gehad. Deze informatie is beperkt en maakt dat er niet verlangd kan worden dat zal worden berekend welke toestand bij de volgende stap precies de beurt krijgt; slechts de kansen om bij de volgende stap de beurt te krijgen kunnen aangegeven worden.

Bij de volgende stap zullen de mogelijke toestanden elk met een bepaalde kans aan bod komen. Deze kansen worden vastgelegd door de voorgeschiedenis. Die bestaat bij Markovketens uit niet meer dan een lijst van alle toestanden die voorafgaand achtereenvolgens aan bod zijn geweest. De belangrijke *essentiële beperking* bij Markovketens is daarom het uitsluitend letten op de stapsgewijze toestandswisselingen, op het slechts oog hebben voor discrete toestandsveranderingen.

De verdere in zekere zin praktische inperking is de eis dat de relevante voorgeschiedenis beperkt is. De kans op de komende toestand moet zijn vastgelegd door een eindige sequentie van voorafgaande toestanden. In de praktijk houdt dit in —want de informatie moet hanteerbaar blijven— dat er geen lange voorgeschiedenis mag meespelen, liefst maar enkele stappen terug: het systeem moet 'kort van memorie' zijn.

Als de relevante voorgeschiedenis n stappen terug gaat, spreken we van een n-staps of *n-de orde* Markovketen. De kans op de komende toestand hangt in zo'n Markovketen af van de sequentie van toestanden, die in de n voorgaande stappen optrad. Men formuleert dit met het begrip *geheugen*: er is een 'geheugen' van n stappen. Het geheugen is bij een Markovketen dus beperkt.

In ons standaardvoorbeeld werd de relevante voorgeschiedenis superkort verondersteld: de schutter schoot raak met een kans van 4/5 als hij zojuist had geraakt en met een kans van 1/2 als hij zojuist had gemist. Kennis over de huidige toestand werd voldoende geacht om de kans op raak bij het volgende schot te geven. Vandaar de uitspraak, dat het systeem uit het voorbeeld helemaal geen geheugen heeft (of een geheugen van één stap, het is maar hoe je het bekijkt).

Bij zulke eerste orde Markovketens worden de overgangskansen compact genoteerd met behulp van een overgangsmatrix **P**, waarin alle kansen op de beurt vanuit de diverse toestanden staan aangegeven. Omdat we natuurlijk willen veronderstellen dat de omstandigheden in de loop van de tijd zich niet essentieel wijzigen, zal deze overgangsmatrix **P** bij elke stap dezelfde zijn.

Het blijkt zelfs dat het voldoende is zich te beperken tot eerste orde Markovketens, zie §10.4.

Als we een gebeuren met een Markovketen gaan beschrijven wordt alle informatie dus vastgelegd in een overgangsmatrix. Zo'n overgangsmatrix heeft een kenmerkende eigenschap: de kentallen, die nooit negatief zijn, hebben in elke rij als som 1. Matrices met deze eigenschap worden *stochastische* matrices genoemd. Dat de som van de overgangskansen vanuit elke toestand 1 is, wordt compact geformuleerd met de kolomvector $\mathbf{I}(n)$, bestaande uit n enen. Voor iedere n-bij-n stochastische matrix **P** geldt

$$\mathbf{P}\,\mathbf{I}(n) = \mathbf{I}(n) \tag{9.1}$$

9.3.1. voorbeeld: geheugen van twee stappen

Als een schutter wat aangeslagen raakt door een mislukt schot, moet zijn presteren misschien wat anders worden beschreven. De kans raak te schieten bij het volgende schot zou kunnen zijn

4/5 als zijn beide voorgaande pogingen raak waren,
3/5 als zijn voorgaande poging raak was en zijn poging daarvoor mis,
1/2 als zijn voorgaande poging mis was en zijn poging daarvoor raak,
2/5 als zijn voorgaande poging mis was en zijn poging daarvoor ook mis.

Deze informatie wordt samengevat in

$$\mathbf{P} = \begin{array}{c|cccc} & rr & rm & mr & mm \\ rr & 4/5 & 1/5 & 0 & 0 \\ rm & 0 & 0 & 1/2 & 1/2 \\ mr & 3/5 & 2/5 & 0 & 0 \\ mm & 0 & 0 & 2/5 & 3/5 \end{array}$$

Dit is de overgangsmatrix voor deze Markovketen met een 'geheugen van 2 stappen'. De uitkomst van twee opeenvolgende schoten is genoteerd als raak-raak (*rr*), raak-mis (*rm*), mis-raak (*mr*) en mis-mis (*mm*).

9.4. ERGODISCHE MARKOVKETENS

9.4.1. limietwaarde toestandsvector

Om in het standaardvoorbeeld de situatie na het n-de schot te voorspellen moeten we de overgangsmatrix **P** tot de n-de macht brengen. Voor de laagste machten van n is

$$\mathbf{P} = \begin{bmatrix} 4/5 & 1/5 \\ 1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} 0.800 & 0.200 \\ 0.500 & 0.500 \end{bmatrix}$$

$$\mathbf{P}^2 = \begin{bmatrix} 37/50 & 13/50 \\ 13/20 & 7/20 \end{bmatrix} = \begin{bmatrix} 0.740 & 0.260 \\ 0.650 & 0.350 \end{bmatrix}$$

$$\mathbf{P}^3 = \begin{bmatrix} 361/500 & 139/500 \\ 139/200 & 61/200 \end{bmatrix} = \begin{bmatrix} 0.722 & 0.278 \\ 0.695 & 0.305 \end{bmatrix}$$

$$\mathbf{P}^4 = \begin{bmatrix} 7166/10000 & 2834/10000 \\ 7085/10000 & 2915/10000 \end{bmatrix} = \begin{bmatrix} 0.717 & 0.283 \\ 0.709 & 0.292 \end{bmatrix}$$

In een ander geval met een overgangsmatrix, waarin drie toestanden worden onderscheiden, is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \end{bmatrix} \quad \mathbf{P}^2 = \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \end{bmatrix} \quad \mathbf{P}^4 = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/4 & 1/4 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

$$\mathbf{P}^8 = \begin{bmatrix} 1/4 & 3/8 & 3/8 \\ 3/16 & 7/16 & 3/8 \\ 3/16 & 3/8 & 7/16 \end{bmatrix} \quad \mathbf{P}^{16} = 1/256 \begin{bmatrix} 52 & 102 & 102 \\ 51 & 103 & 102 \\ 51 & 102 & 103 \end{bmatrix}$$

In beide gevallen blijken de matrices $\mathbf{P}^n$ bij toename van n steeds wat meer op elkaar te gaan lijken. Als we de berekeningen veel verder voortzetten —we laten de computer dit werk doen— constateren we dat deze trend behouden blijft als n

groter en groter wordt. De verschillen tussen $\mathbf{P}^n$ en $\mathbf{P}^{n+1}$ worden miniem voor grote waarden van n; het maakt dan weinig meer uit welke waarde n precies heeft.

We kunnen met veel matrices $\mathbf{P}$ experimenteren en vinden vaak dat $\mathbf{P}^n$ voor grote waarden van n nog maar weinig met n varieert. Op grond van deze waarneming vermoeden we dat $\mathbf{P}^n$ in zulke gevallen een limietwaarde heeft voor n naderend tot oneindig. Het vermoeden blijkt juist te zijn voor een belangrijke algemene klasse van stochastische matrices $\mathbf{P}$. Die matrices worden *ergodisch* genoemd, ze beschrijven *ergodische Markovketens*. Voor iedere ergodische overgangsmatrix bestaat de limiet van $\mathbf{P}^n$ voor n naar oneindig.

De overgangsmatrix uit het standaardvoorbeeld is zo'n ergodische matrix, in het volgende hoofdstuk zien we in §10.3.1 dat de limiet bestaat. Ook de stochastische matrix in het andere voorbeeld is ergodisch. We bepalen ons in dit hoofdstuk tot de ergodische ketens; in het volgende hoofdstuk bekijken we Markovketens in het algemeen.

De limiet voor $\mathbf{P}^n$ geven we steeds aan met de matrix Π. Daarmee bedoelen we: het element $\mathbf{P}^n_{ij}$ (uit de n-de macht van $\mathbf{P}$ het element in de i-de rij en de j-de kolom) wordt als n maar groot genoeg is vrijwel Π_{ij}: het element in de i-de rij en de j-de kolom van de limietmatrix Π.

De limieteigenschap van de matrices $\mathbf{P}^n$ maakt dat ook de toestandsvector op den duur, voor grote waarden van n, steeds vrijwel dezelfde is. In het standaardvoorbeeld blijkt bijvoorbeeld dat de toestandsvector na het 100-ste schot is†

$$(p_{100}(0), p_{100}(1)) \quad = (1,0)\,\mathbf{P}^{100} \approx (1,0)\,\Pi = (\pi(0), \pi(1))$$

en na het 1000-ste schot

$$(p_{1000}(0), p_{1000}(1)) = (1,0)\,\mathbf{P}^{1000} \approx (1,0)\,\Pi = (\pi(0), \pi(1))$$

Ons model voorspelt hier dat de kans om bij het 100-ste schot raak te schieten (vrijwel) even groot is als de kans om bij het 1000-ste schot raak te schieten. Er wordt voor grote waarden van n met kansen van $(\pi(0), \pi(1))$ raak en mis geschoten. Er hoeft niet meer aangegeven te worden hoe groot n precies is, mits 'groot'.

Het blijkt aanstonds dat de toevoeging dat er bij het nulde schot is geraakt, voor grote waarden van n ook niet meer nodig is. Als bij het nulde schot zou zijn gemist is de toestandsvector na het n-de schot voor grote waarden van n gelijk aan $(0,1)\,\Pi$ en dat blijkt gelijk te zijn aan $(1,0)\,\Pi$! Informatie over wat er bij het nulde schot gebeurde heeft blijkbaar voor voorspellingen op de langere termijn geen enkele waarde.

---

† 'Ongeveer gelijk aan' wordt weer aangegeven door $\approx$.

De voor het standaardvoorbeeld aangegeven resultaten gelden algemeen voor de grote klasse van de ergodische Markovketens. De limietwaarde van de toestandsvector is onafhankelijk van de begintoestand. De toestandsvectoren na de n-de stap hangen vrijwel noch van n, noch van de begintoestandsvector af. De limietwaarde noemen we de *stationaire toestandsvector*, aangeduid met π. Men formuleert ook dit met het begrip geheugen en constateert dat er een beperkt geheugen is: de kans op een bepaalde toestand is na de k-de stap even groot als na de m-de stap, mits k en m maar groot genoeg zijn. Deze kans hangt alleen af van de kansen om van de ene naar de andere toestand te gaan, dus van de overgangsmatrix; de begintoestand doet er niet toe. Hoe groot k en m precies moeten zijn hangt natuurlijk af van de gezochte precisie van de uitspraak; wat 'groot' is, is altijd relatief.

9.4.2. het begrip stationair

Wat hier zo simpelweg wordt vastgesteld is uiterst belangrijk. Bij Markovketens blijkt het mogelijk te zijn een consistente definitie te geven van het alledaagse begrip *voorspelling op de lange termijn.* In het dagelijks spraakgebruik is deze uitdrukking nuttig, maar wat vaag. In de context van ergodische Markovketens kunnen we precies aangeven, wat er onder zal worden verstaan: de limietwaarde (voor n groot) van de toestandsvector. Deze limietwaarde is onafhankelijk van de begintoestandsvector, dat is nodig om de zegswijze 'op de lange termijn' zonder nadere toevoegingen te kunnen gebruiken.

Men drukt zich echter binnen het vakgebied vaak op een andere manier over dit fenomeen uit. Er wordt gezegd dat het systeem op den duur *stationair* wordt, of in *evenwicht* komt; het systeem raakt in de stationaire fase. Deze manier van zeggen kan de niet-ingewijde in verwarring brengen. Stationair worden wil hier beslist niet zeggen dat alle veranderingen 'uitdempen' en het systeem op den duur vrijwel continu in dezelfde toestand verkeert. Integendeel, het wil niets meer en niets minder zeggen dan dat het mogelijk wordt zonder meer over kansen op een bepaalde toestand te spreken. In geval van twijfel aan de bedoeling van de uitdrukking *stationaire toestand* of *evenwichtstoestand* is het altijd de beste remedie terug te gaan naar de primaire betekenis: de kans op de toestand in de stationaire fase is de kans op die toestand na de n-de stap, met n groot.

De kans om in het standaardvoorbeeld bij het 1000-ste schot over te gaan van de toestand mis op de toestand raak, is $p_{999}(1)\times 1/2$. De stationaire toestandsvector wordt in de volgende paragraaf bepaald op $(\pi(0), \pi(1)) = (5/7, 2/7)$. De kans om een gemist schot te laten volgen door een raak schot is dus vrijwel $2/7\times 1/2 = 1/7$. De specificatie 'bij het 1000-ste schot' doet er weer niet toe. Er is bij de schutters in de limiet een kans van 1 op 7 om te switchen van mis naar raak.

Bij alle ergodische ketens zijn de kansen op bepaalde toestandsovergangen in de stationaire fase onafhankelijk van het stapnummer. Er vinden overgangen tussen toestanden plaats met *'vaste kansen'*. Het is deze eigenschap die de naamgeving *stationair*, dat is *'in evenwicht'*, heeft doen ontstaan.

9.4.3. *voorbeeld: wedstrijdjes*

We laten twee teams, beide bestaande uit 7 van onze schutters, tegen elkaar spelen. Elke schutter mag één keer schieten. Het team met de meeste raakschietende schutters wint. Voor de 7 schutters in team 'wit' nemen we schutters, die zojuist geraakt hebben; voor de 7 schutters in team 'rood' nemen we 7 schutters, die zojuist gemist hebben.

Als we vaak zulke teams formeren, constateren we vaker dat wit wint dan rood. Bij wit schieten immers gemiddeld $4/5 \times 7$ schutters raak, bij rood $1/2 \times 7$. Laten we dezelfde teams echter meer partijen achterelkaar tegen elkaar spelen, dan vervaagt dit verschil. Bij beide teams schieten na enige tijd vrijwel gemiddeld $\pi(0) \times 7 = 5$ schutters raak. Nu eens wint wit, dan weer rood, al naar het noodlot het uitmaakt en beide teams winnen gemiddeld vrijwel even vaak. De aanvankelijke handicap voor rood, dat alle schutters gemist hebben, telt op de lange termijn niet omdat volgens de veronderstellingen van het eerste orde Markovmodel de handicap is verdwenen zodra de betrokken schutter eenmaal geraakt heeft.

9.5. BEPALING VAN DE STATIONAIRE TOESTANDSVECTOR

9.5.1. *evenwichtsvergelijking*

We willen nagaan hoe we de stationaire toestandsvector kunnen bepalen. Als

$$\mathbf{P}^{n+1} \approx \mathbf{P}^n \approx \Pi$$

moet gelden

$$\Pi \mathbf{P} = \mathbf{P} \Pi = \Pi$$

De toestandsvectoren na het n-de en het (n + 1)-ste schot zijn gelijk aan

$$p_n = p_0 \mathbf{P}^n; \qquad p_{n+1} = p_0 \mathbf{P}^{n+1}$$

Voor grote n zouden de beide toestandsvectoren vrijwel gelijk zijn aan de stationaire toestandsvector π met

$$\pi = p_0 \Pi$$

Hier staat dat de matrix Π, de limietwaarde voor n naar oneindig van $\mathbf{P}^n$, de overgangsmatrix is naar de stationaire toestand.

De voorgaande betrekkingen laten samen zien dat de stationaire toestandsvector moet voldoen aan de volgende 'evenwichts'relatie

$$\pi = \pi \mathbf{P} \tag{9.2}$$

of†

$$\pi(\mathbf{I}-\mathbf{P}) = 0 \tag{9.3}$$

Als er n verschillende toestanden zijn, is **P** een n-bij-n overgangsmatrix en zijn dit n vergelijkingen. Deze belangrijke vergelijkingen, die de kern aangeven van de ergodische Markovketens, worden de *evenwichtsvergelijkingen* genoemd. Om de stationaire toestandsvector te bepalen is het voldoende het stelsel evenwichtsvergelijkingen op te lossen.

9.5.2. stationaire kansen in standaardvoorbeeld

In het standaardvoorbeeld met de overgangsmatrix **P** uit figuur 9.1 houdt de evenwichtsrelatie $\pi = \pi\mathbf{P}$ de volgende twee vergelijkingen in:

$$\pi(0) = 4/5\,\pi(0) + 1/2\,\pi(1)$$
$$\pi(1) = 1/5\,\pi(0) + 1/2\,\pi(1)$$

of in de vorm $\pi(\mathbf{I}-\mathbf{P}) = 0$:

$$1/5\ \pi(0) - 1/2\ \pi(1) \quad = 0$$
$$-1/5\ \pi(0) + 1/2\ \pi(1) = 0$$

De twee evenwichtsvergelijkingen zijn afhankelijk en we vinden niet meer dan $\pi(0)/\pi(1) = 5/2$. Aangezien $\pi(0)$ en $\pi(1)$ kansen zijn met als som 1, geldt 'gelukkig' bovendien

$$\pi(0) + \pi(1) \ = 1$$

Dus is $\pi(0) = 5/7$ en $\pi(1) = 2/7$. Hiermee is de vergelijking voor de stationaire toestandsvector opgelost. De stationaire toestand wordt beschreven door de toestandsvector (5/7, 2/7).

Er is voor grote n, 'op den duur', een kans van 5/7 dat raak en een kans van 2/7 dat mis geschoten wordt. Hoe groot 'grote n' precies zal zijn hangt af van de precisie waarmee men de kansen wil kennen. We zullen in het volgende hoofdstuk

† **I** is de eenheidsmatrix.

een methode bespreken, waarmee de snelheid waarmee de limiet genaderd wordt kan worden bepaald (§10.3.1).

De overgangen raak-raak, raak-mis, mis-raak en mis-mis vinden in de stationaire toestand plaats met kansen $\pi(0)\times 4/5$, $\pi(0)\times 1/5$, $\pi(1)\times 1/2$, $\pi(1)\times 1/2$. Dus met de kansen 4/7, 1/7, 1/7 en 1/7.

9.5.3. *oplossing evenwichtsvergelijkingen algemeen*

De oplossing van de evenwichtsrelaties loopt altijd zoals in het standaardvoorbeeld. Het stelsel vergelijkingen $\pi(\mathbf{I}-\mathbf{P}) = 0$ is steeds afhankelijk, omdat voor stochastische matrices (§9.3) $(\mathbf{I}-\mathbf{P})\mathbf{I}(n) = \mathbf{I}(n) - \mathbf{I}(n) = 0$. De stationaire kansen zijn dus slechts op een factor na uit de evenwichtsvergelijking te bepalen; ze worden eenduidig vastgelegd door de eis dat de som van de kansen op de mogelijke toestanden steeds, en dus ook in de limiet, 1 moet zijn.

De vergelijkingen $\Pi = \Pi\mathbf{P} = \mathbf{P}\Pi$ kunnen op dezelfde manier opgelost worden. In het standaardvoorbeeld heeft de overgangsmatrix naar de stationaire toestand de vorm

$$\Pi = \begin{bmatrix} \pi_{00} & \pi_{01} \\ \pi_{10} & \pi_{11} \end{bmatrix}$$

Voor de eerste rij van Π geldt, omdat $\Pi=\Pi\mathbf{P}$

$$(\pi_{00}, \pi_{01}) = (\pi_{00}, \pi_{01})\mathbf{P}$$

Deze relatie is precies dezelfde als $(\pi(0), \pi(1)) = (\pi(0), \pi(1))\mathbf{P}$ voor de stationaire toestandsvector $(\pi(0), \pi(1))$. Bovendien geldt ook dat $\pi_{00}+\pi_{01} = 1$. De reden daarvoor is dat $\mathbf{P}^n$ voor iedere n een overgangsmatrix is, zodat de overgangskansen vanuit een bepaalde toestand, die samen een rij vormen uit $\mathbf{P}^n$, als som 1 hebben. Deze eigenschap blijft in de limiet behouden, dus de som van de elementen uit de eerste rij van Π is 1. De enige mogelijke conclusie is dat de eerste rij van Π gelijk is aan de stationaire toestandsvector $(\pi(0), \pi(1))$!

Als we $\Pi = \Pi\mathbf{P}$ verder uitschrijven, vinden we voor de tweede rij

$$(\pi_{10}, \pi_{11}) = (\pi_{10}, \pi_{11})\mathbf{P}$$

Dit is weer helemaal dezelfde relatie en om dezelfde redenen is

$$(\pi_{10}, \pi_{11}) = (\pi(0), \pi(1))$$

We constateren dat de rijen van de matrix Π identiek zijn:

$$\Pi = \begin{bmatrix} \pi(0) & \pi(1) \\ \pi(0) & \pi(1) \end{bmatrix}$$

Voor een algemene ergodische overgangsmatrix **P** zal dit op precies dezelfde manier gevonden worden. De rijen van Π zijn dus altijd gelijk aan de stationaire toestandsvector.

Het feit dat de rijen van Π gelijk zijn, maakt dat de begintoestand niet relevant is voor de stationaire toestand. Omdat voor elke p_0 de som van de elementen 1 is en de rijen van Π gelijk zijn, is altijd

$$p_0 \Pi \quad = \pi$$

(en inderdaad $\mathbf{P}\Pi = \Pi$).

In §9.4.1 zijn de stationaire toestandsvectoren van de voorbeelden respectievelijk (5/7, 2/7) en (1/5, 2/5, 2/5). We zien daar inderdaad dat de rijen in $\mathbf{P}^3$, $\mathbf{P}^4$ en $\mathbf{P}^8$, $\mathbf{P}^{16}$ op π gaan lijken.

9.6. BETEKENIS VAN DE STATIONAIRE TOESTAND

We moeten als we de uitkomsten voor de kansen in de stationaire fase willen toepassen, altijd duidelijk onderscheid maken tussen voorspelling en informatie.

Wat verwacht de commandant als hij naar een oefening gaat kijken en zo maar een bepaalde schutter, die op het punt staat te gaan schieten, in het oog vat? Als de commandant over dezelfde ervaring beschikt als een geroutineerde schutter zou hij, onze berekening volgend, moeten verwachten dat de schutter met een kans van $(1,0)\Pi = (5/7, 2/7)$ of $(0,1)\Pi = (5/7, 2/7)$ raak schiet of mist, kortom met een kans van 5 op 7 raakt en van 2 op 7 mist. Hij weet immers niets meer van de schutter dan dat hij eens is begonnen met schieten (raak of niet raak, het doet er niet toe) en daarna er mee door is gegaan en nu op het punt staat opnieuw te schieten (n-de schot, *n* heel groot).

Als de commandant merkt dat de schutter raak schiet wijzigt hij op grond van deze nieuwe informatie zijn voorspelling voor het dan volgend schot: het zal raak zijn met een kans van 4/5 en mis met een kans van 1/5 (de kans op raak is inderdaad groter dan zojuist).

Aan een koning, die ook wel eens komt kijken, dient hij te melden: Verwacht, als U zo maar lukraak een afgaand schot volgt, raak met een kans van 5/7 en mis met een kans van 2/7. Maar als U iemand hebt zien missen, verwacht dan bij de volgende poging in de helft van de gevallen weer mis.

Waarschijnlijk zal de commandant ook rapporteren dat een schutter gemiddeld in 5/7×100% = 71,4% van de gevallen raak schiet. Hij bepaalt dan het relatief aantal treffers met de kans 5/7 op raak, een kans die geldt in de evenwichtstoestand, en doet dus alsof de kans op raak steeds de kans is voor de stationaire toestand. Dit lijkt wel handig, maar is het logisch correct? We gaan er nader op in.

9.6.1. over/op de lange termijn

Bij het eerste schot van een schutter is de kans op raak/mis $p_1 = p_0\mathbf{P}$, bij het k-de schot $p_k = p_0\mathbf{P}^k$. De kansen op raak zijn van schot tot schot verschillend. Van alle k-de schoten zal gemiddeld een deel p_k raak/mis zijn. Er kan zo worden aangegeven welk deel van ieders schoten tot en met ieders N-de schot gemiddeld raak of mis zal zijn; dit staat in

$$\sum_{k=1}^{N} p_k/N = p_0 \sum_{k=1}^{N} \mathbf{P}^k/N$$

Voor de som van de machten van **P** kan een benadering worden gegeven, die beter is naarmate de bovengrens N hoger ligt: bij benadering geldt de matrixgelijkheid

$$\sum_{k=1}^{N} \mathbf{P}^k/N = \Pi$$

zodat het deel raak/mis van de schoten is

$$p_0\Pi = \pi$$

De reden hiervan is de volgende. Voor zeg $k > K$ verschilt $\mathbf{P}^k$ niet meer 'merkbaar' van zijn limietwaarde Π. Splits de som daarom in twee stukken, in het eerste part is nog niet $\mathbf{P}^k$ vrijwel Π, in het tweede part wel. Dus

$$\sum_{k=1}^{N} \mathbf{P}^k/N \approx \sum_{k=1}^{K} \mathbf{P}^k/N + \Pi(1-\frac{K}{N})$$

De som rechts is een eindige som van termen, gedeeld door N. Voor grote N is dat maar een kleine bijdrage. De term $\Pi(1-K/N)$ is als N groot is vrijwel Π. Als de N maar groot genoeg is ten opzichte van K overheerst in het rechterlid de eerste bijdrage uit de laatste term en is de uitkomst vrijwel Π.

Dit is een wiskundig uiterst slordig betoog. Gelukkig zijn de eigenschappen van de gemiddelde som beter onderzocht. Heel algemeen is gebleken dat het aanloopstuk (het stuk zeg tot en met de K-de stap) een bijdrage levert die voor grote waarden van N relatief steeds minder betekent. Een belangrijke stelling uit de

theorie van de Markovketens zegt dat voor ergodische Markovketens

$$\lim_{N\to\infty} \sum_{k=1}^{N} \mathbf{P}^k / N = \Pi$$

zodat

$$\lim_{N\to\infty} p_0 \sum_{k=1}^{N} \mathbf{P}^k / N = p_0\Pi = \pi$$

Dit algemene resultaat kunnen we zo parafraseren: de kans *over* de lange termijn (het linkerlid van de gelijkheid) is de kans *op* de lange termijn (het rechterlid van de gelijkheid).

Op basis van deze stelling mag de commandant over zijn schutters opmerken dat gemiddeld over ieders pogingen bij de komende schoten, en daarmee over de hele groep schutters te zamen, er gemiddeld relatief raak/mis zal worden geschoten overeenkomstig π. Dat deed hij eigenlijk toen hij rapporteerde dat een schutter in 71,4% van de gevallen raak schiet.

Het onderscheid in een 'aanloopfase' of 'opwarmfase' (vakterm: *transient state*) —hier $k \leqslant K$— en een stationaire fase of evenwichtsfase —hier $k > K$— is heel wezenlijk, men komt het telkens weer tegen. In wachttijdensystemen is de aanloopfase de tijd direkt na het aan het werk gaan van de server. In simulaties gaat het om de eerste resultaten uit een lange simulatierun. Bij het onderzoek aan page faults bij gepagineerd geheugenbeheer is het het binnenhalen van de workingset (de 'koude start' van het pagineringsmechanisme).

9.7. VOORBEELD: PENDELEND SYSTEEMPROCES

Tot een zeker operatingsysteem behoort ook een bepaald systeemproces Progressor, dat zo nu en dan een bepaalde taak verricht. Dit systeemproces kan zich in drie fasen bevinden: runnable, sleeping en idle; dit zijn de drie toestanden *runnable*, *sleeping* en *idle*.

Zodra Progressor door een systeeminterrupt gewekt is uit de toestand idle wordt het runnable; één van de CPU's gaat het systeemproces Progressor verwerken, zo gauw het aan de beurt is. Het executeren van Progressor resulteert in gemiddeld 2 op de 3 gevallen in een opdracht aan één van de randapparaten om bepaalde gegevens aan te leveren; in de andere gevallen wordt de executie van Progressor gestaakt, omdat de opgegeven taak geklaard is, de toestand wordt idle. Progressor gaat zodra er I/O moet worden gepleegd over in de toestand sleeping. Als de I/O-verwerking klaar is blijkt in gemiddeld de helft van de gevallen de taak van Progressor vooreerst afgelopen, de toestand wordt idle. In de andere helft van de gevallen moet de executie voortgezet worden en wordt Progressor weer runnable.

Progressor pendelt zo tussen twee bezoeken aan de toestand idle een aantal malen heen en weer tussen de toestanden *runnable* en *sleeping*, hierbij wordt het

afwisselend behandeld door één van de CPU's en één van de randapparaten, in die tijd is het 'aktief'.

De toestandsovergangen van Progressor kunnen beknopt worden weergegeven in een overgangsmatrix. Met toestand_1 = idle, toestand_2 = runnable en toestand_3 = sleeping is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 1/3 & 0 & 2/3 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

We hebben dan een vereenvoudiging aangebracht, die niet strikt uit de gegeven specificaties volgt. In de beschrijving van het gebeuren staat voor elke overgang hoe vaak die overgang gemiddeld optreedt. In de overgangsmatrix staan vaste overgangskansen voor overgangen. Dat veronderstelt meer.

Volgens de overgangsmatrix vindt bijvoorbeeld aan het eind van elke fase 'runnable' na de 'CPU-burst' (periode van ononderbroken executie) met een vaste kans van 1 op 3 een overgang naar 'idle' plaats; deze kans zou niet veranderen tijdens het aktief zijn van Progressor. Misschien is de kans in werkelijkheid echter kleiner dan 1 op 3 aan het begin van een aktieve periode, bij de eerste keer runnable mogelijk niet 1 op 3 maar 1 op 4, en is ze bij de volgende keren groter, bij de vierde keer bijvoorbeeld niet 1 op 3 maar 1 op 2. Gemiddeld kan dan best in 1 op de 3 gevallen de overgang worden gemaakt. In de overgangsmatrix wordt simpelweg de kans vast verondersteld en voor de vaste kans de gespecificeerde waarde, waarin over het totale verloop gemiddeld is, genomen.

Zolang verdere informatie ontbreekt doen we er verstandig aan vooreerst met vaste kansen te rekenen, en dus bijvoorbeeld een vaste kans van 1 op 3 voor de overgang naar idle bij elk einde van de toestand runnable te nemen. Dit is te rechtvaardigen met een van de volgende argumenten, of een combinatie van beide:

- er zijn goede redenen te veronderstellen dat de kansen, zo ze al uiteenlopen, niet sterk verschillen. Een argument kan zijn dat de code van Progressor resident is en cyclisch wordt doorlopen, waarbij een 'idle' toestand overal in de code kan optreden.
- analyse van een uitgebreider model, waarin de kansen wel verschillen, bijvoorbeeld een n-de orde Markovketen, laat zien dat het voor de globale uitspraken, waarvoor het model bedoeld is, weinig uitmaakt hoe de spreiding in de kansen over een aktieve periode van Progressor precies is, als ze maar '*gemiddeld*' 1 op 3 zijn, wat we in de overgangsmatrix inbrengen door voor de overgangskans 1/3 te nemen. Probleem blijft dan een beetje hoe er 'gemiddeld' wordt. Zie daarvoor straks ook de operationele benadering in het hoofdstuk MARKOVKETENS EN EVENWICHTSVERGELIJKINGEN.

Hier ligt een van de taken van de 'theoretische' performance-analyse: aan te geven hoe '*robuust*' de modellen zijn. Markovmodellen zijn robuust gebleken.

We rekenen verder met vaste kansen. De stationaire toestandsvector die bij de overgangsmatrix **P** hoort, vinden we uit (9.3)

$$\pi(\mathbf{I}-\mathbf{P}) = 0$$

of met $\pi = (\pi_1, \pi_2, \pi_3)$

$$\begin{aligned} \pi_1 - 1/3\pi_2 - 1/2\pi_3 &= 0 \\ -\pi_1 + \pi_2 - 1/2\pi_3 &= 0 \\ -2/3\pi_2 + \pi_3 &= 0 \end{aligned}$$

met als oplossing $\pi_1 = \pi_3, \pi_2 = 3/2\pi_3$ en omdat $\pi_1 + \pi_2 + \pi_3 = 1$

$$(\pi_1, \pi_2, \pi_3) = (2/7, 3/7, 2/7)$$

Een systeemproces als Progressor is in de toestanden *idle, runnable* of *sleeping*. Als Progressor gevolgd wordt vanaf de toestand idle, zal het proces na 1000 overgangen tussen idle, runnable en sleeping met een kans van $(1,0,0)\mathbf{P}^{1000}$ in de respectievelijke toestanden idle, runnable en sleeping zijn. Deze kansen zullen vrijwel gelijk zijn aan (2/7, 3/7, 2/7) (als 1000 'voldoende' groot is). Als niet vanaf de toestand idle wordt gerekend, maar vanaf de toestand runnable blijven deze kansen dezelfde.

Iedere toestand correspondeert met een fase. Op de lange duur maakt het proces tweemaal de fase idle door tegen driemaal de fase runnable en tweemaal de fase sleeping. In de evenwichtstoestand is het proces per verblijf in de fase idle gemiddeld 3/2 maal runnable en eenmaal sleeping. Van de drie fasen idle, runnable en sleeping komt over de lange termijn gezien de fase idle even vaak voor als de fase sleeping en de fase runnable komt anderhalf maal zo vaak voor. De drie fasen hebben de beurt in de verhouding 2:3:2. De kansen 2/7, 3/7 en 2/7 zijn kennelijk *kansen op de beurt*, het zijn *kansen op optreden* van toestanden.

9.8. OVERALL-BEELD ERGODISCHE MARKOVKETEN

We geven de resultaten voor ergodische Markovketens nog eens in een formulering met beurten.

Bij Markovketens bestaat de ingebrachte informatie slechts uit de kansen waarmee de verschillende toestanden vanuit de mogelijke toestanden de beurt krijgen. Deze informatie staat in de overgangsmatrix **P**. Uit deze informatie worden met de evenwichtsvergelijking $\pi(\mathbf{I}-\mathbf{P})=0$ de kansen op de beurt voor de diverse toestanden in de stationaire toestand bepaald. Deze kansen geven ook aan hoe vaak over de lange termijn gezien de diverse toestanden de beurt hebben. Dat is het hart van de theorie van ergodische Markovketens.

De veronderstelling dat de omstandigheden blijvend gelijk zijn is bij Markovketens dat bij elke stap dezelfde matrix **P** geldt.

Het is belangrijk te beseffen dat de kansen volgens de stationaire toestandsvector π overeenkomen met wat we in het hoofdstuk BEURTEN EN KANSEN onder de kans op optreden —de kans op de beurt— verstonden. De kansen uit π zijn te bepalen door te tellen. Bij Markovketens hebben we een situatie in handen, die valt binnen de heel algemeen gehouden omschrijving in 'beurten en kansen'. De veronderstelling dat een ergodische Markovmodel de situaties beschrijft is een bepaalde specificatie van de daar aangegeven situatie. Alle resultaten uit dat hoofdstuk zijn dus van toepassing.

Nu blijkt heel duidelijk dat er in BEURTEN EN KANSEN bij de kans op optreden niet ook maar impliciet wordt verondersteld dat opeenvolgende beurten te zien zouden zijn als het resultaat van opeenvolgende onafhankelijke trekkingen met een kans gelijk aan de kans op de beurt, dus volgens een binomiale verdeling. Dergelijke onafhankelijke opeenvolgende trekkingen met steeds dezelfde kans op succes noemt men Bernouilli-trekkingen; het systeem ontwikkelt zich daarbij als een *Bernouilli-proces*. Bij Markovketens bestaat tussen opeenvolgende toestanden een verband volgens de matrix **P**, en dat is een heel ander verband dan bij telkens lukraak de volgende toestand trekken met kansen volgens de 'overall' kansen uit π. Markovketens zijn aanmerkelijk rijker dan Bernouilli-processen.

Verdergaande resultaten van de Markovtheorie voor ergodische ketens zijn op het eerste gezicht voor de hand liggend, maar bij verder nadenken eigenlijk verrassend. We geven geen afleidingen meer en verwijzen naar leerboeken over Markovketens en naar het voorbeeld in de volgende paragraaf.

Laten we de kans in de stationaire toestand dat X optreedt, dat toestand$_X$ de beurt heeft, aangeven met π_X. Het gemiddeld aantal stappen of beurten tussen twee opeenvolgende malen dat X de beurt heeft, blijkt gelijk te zijn aan $1/\pi_X - 1$, het aantal stappen of beurten vanaf de laatste beurt tot en met de volgende keer dat X de beurt heeft is $1/\pi_X$. Het is 'net alsof' de beurten getrokken worden volgens een binomiale verdeling met kans π_X op de beurt aan X, $1-\pi_X$ op de beurt niet aan X, maar dat is echt niet het geval.

Het gemiddeld aantal malen dat toestand Y de beurt heeft in zo'n tussenpoos tussen twee X-en in, is

$$n_Y = \pi_Y / \pi_X$$

Weer alsof gemiddeld $1/\pi_X$ maal wordt getrokken volgens een binomiale verdeling met kans π_Y op succes.

De verhouding van de π's geeft dus aan hoe vaak de diverse toestanden relatief optreden, wat de relatieve frequenties zijn waarmee ze worden aangedaan, kortom: de π's beschrijven de verdeling van de toestanden in de stationaire fase.

9.9. Markovketens en computerconfiguraties

Een belangrijke toepassing van de Markovketens ligt bij de analyse van verwerkingsduren van transakties (programma's) bij devices in computerconfiguraties. Voor het verkrijgen van meetgegevens wordt gebruik gemaakt van softwaremonitoren die de gang van zaken binnen het operatingsysteem via tellers bijhouden. We gaan weer uit van een illustratief voorbeeld. De stationaire toestandsvector π wordt gebruikt om de verwerkingsduren bij de devices te bepalen. Deze verwerkingsduren leggen vast welk device *knelpunt* is.

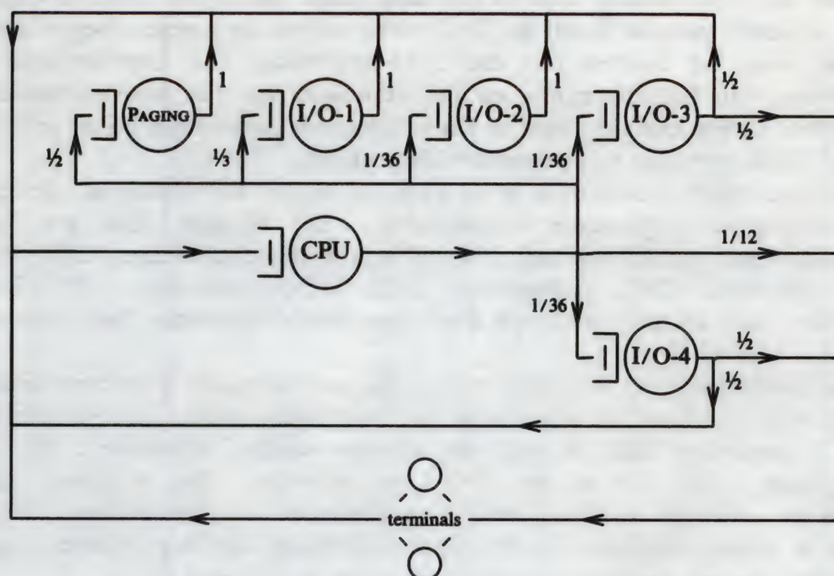
9.9.1. voorbeeld: accessen per usercommando

Een computersysteem verwerkt usercommando's onder een operatingsysteem, dat gebruik maakt van het 'virtuele geheugen' concept. Het executeren van een usercommando wordt verzorgd door de centrale verwerkingseenheid, de CPU. De verwerking van page faults (zie bijvoorbeeld het voorbeeld §10.5 uit het volgende hoofdstuk) vindt plaats via de 'pagingdrum' (ook een solid state device noemen we gladweg een drum). De verdere relevante randapparaten zijn 2 disks (I/O-1 en I/O-2) en 2 drums (I/O-3 en I/O-4), die de verwerking van I/O-opdrachten verzorgen (zie figuur 9.4).

Het executeren van een usercommando wordt soms onderbroken voor de afwikkeling van een page fault (behandeling door de pagingdrum), waarna de CPU direkt doorgaat met een ander usercommando, zo dat aanwezig is. Net zo wordt gehandeld bij een onderbreking ter afwikkeling van I/O-verwerking door één van de I/O-randapparaten 1, 2, 3 en 4. Als een acces van een usercommando op een randapparaat is afgewerkt wordt dit usercommando door de CPU weer in executie genomen, zodra het 'aan de beurt' is. Na de afwikkeling van een I/O-3 opdracht blijkt echter gemiddeld in de helft van de gevallen dat de totale verwerking van het usercommando klaar is, idem bij I/O-4 opdrachten, maar de totale verwerking van een usercommando eindigt nooit na een I/O-1 of I/O-2 opdracht.

De verwerking door de CPU van een usercommando kan dus om diverse redenen worden gestaakt. In gemiddeld de helft van de gevallen is de reden het optreden van een page fault, in gemiddeld een derde deel van de gevallen een I/O-opdracht voor randapparaat I/O-1 en in telkens gemiddeld 1/36 deel van de gevallen een I/O-opdracht voor de randapparaten I/O-2, 3 en 4. In de rest van de gevallen (1/12 deel) blijkt het usercommando afgewerkt te zijn, de gebruiker krijgt een melding en zal weldra een nieuw usercommando inbrengen (zo'n terugkeer naar de 'denk'fase is dus alleen onmiddellijk na verwerking door CPU, randapparaat I/O-3 en I/O-4 mogelijk). Alle CPU-aktiviteiten voor systeemfunkties worden bij de verwerking van usercommando's door de CPU ingesloten. Tenslotte is nog bekend dat ieder usercommando begint met een verwerking door de CPU.

Deze gegevens bevatten informatie over de levensloop van een usercommando. Ze beschrijven de toestanden, waarin een usercommando kan verkeren, in termen van *bij de CPU, bij de pagingdrum, bij I/O-1, I/O-2, I/O-3 en I-O-4.* Bij de CPU is een beknopte formulering voor *wordt verwerkt door de CPU of wacht op verwerking*



Figuur 9.4. Pendelende usercommando's.

door de CPU en ook dat is een gestileerde weergave van wat er zich afspeelt. De scheduling bij de CPU bepaalt hoe de zaken bij de CPU-verwerking precies geregeld worden. Bij de ROUND ROBIN (PS) regeling bijvoorbeeld worden usercommando's telkens gedurende een korte time slice verwerkt, het is net alsof ze continu geholpen worden door de CPU, die zijn verwerkingscapaciteit verdeelt over de commando's 'bij de CPU' (§14.10). Bij een FCFS afwikkeling wordt precies één van de usercommando's 'bij de CPU' door de CPU geholpen, de rest wacht. Merk op dat onderbrekingen van het executeren van een usercommando die het gevolg zijn van de scheduling bij de CPU (zoals aan het eind van een time slice) niet als breekpunten in de levensloop van een usercommando worden meegenomen. Bij dat soort onderbrekingen blijft het usercommando in dezelfde toestand, het blijft in de fase 'bij de CPU'. De uitdrukkingen 'bij de pagingdrum' en 'bij een I/O-apparaat' moeten net zo worden geïnterpreteerd.

In de beschrijving van de levensloop komt de individualiteit van het usercommando niet voor. Die is hier niet relevant, op dit niveau van de specifikatie tellen alleen maar verdelingsfunkties met gemiddelde waarden. Elk usercommando zal een eigen weg volgen, maar gemiddeld genomen liggen de overgangsfrequenties als aangegeven. Soms zitten er misschien 4 usercommando's in het systeem, soms wellicht 20, af en toe geen, ook over al dat soort situaties zijn de gegevens gemiddeld.

We kunnen het afgewerkt raken van een usercommando daardoor ook weer zien als een toestandsverandering, het commando komt om zo te zeggen in de toestand *bij de gebruiker*, zoals het ook in de toestand 'bij de CPU' kan zijn. Het inbrengen

van een volgende opdracht door de gebruiker houdt slechts een toestandswisseling in, de volgende toestand is 'bij de CPU' (want volgens de gegevens begint de verwerking door het systeem met een CPU-verwerking). Zo opgevat wordt de beëindiging van het commando en het inbrengen van het nieuwe commando beschreven als een overgang naar en het verlaten van de toestand 'bij de gebruiker' (de tijd bij de gebruiker doorgebracht is de denktijd).

Iedere gebruiker is natuurlijk in de loop van de tijd verbonden met een reeks achtereenvolgens ingebrachte commando's. En daarmee met een reeks toestandswisselingen, bijvoorbeeld ('bij de' is weggelaten) gebruiker, CPU, I/O-1, CPU, gebruiker, CPU, pagingdrum, CPU, I/O-3, gebruiker, CPU, I/O-2, enzovoort. Tot en met I/O-2 zijn drie commando's ingebracht, het derde commando is 'bij I/O-2'.

Deze beschrijving van de gebruiker met zijn usercommando is helemaal analoog aan de beschrijving van de schutter uit het standaardvoorbeeld. De schutter kent maar 2 toestanden: raak en mis; een usercommando 7 toestanden: bij CPU, pagingdrum, I/O-1 tot en met I/O-4 en gebruiker. Een wisselend aantal gebruikers —dus een wisselend aantal usercommando's— correspondeert met een wisselend aantal schutters. Bij de responsetijdrelatie voor een wisselend aantal gebruikers kwamen we dezelfde beschrijvingswijze ook al tegen (§5.6).

De gang van zaken voor de usercommando's binnen zo'n beschrijving wordt weergegeven door de overgangsmatrix, die beknopt beschrijft wat de uitgebreide formulering-in-woorden vertelt (figuur 9.5). Net als bij de overgangsmatrix voor Progressor gaan we uit van vaste kansen, voor deze kansen hebben we de opgegeven gemiddelde waarden genomen.

	CPU	paging	I/O-1	I/O-2	I/O-3	I/O-4	gebruiker
CPU	0	1/2	1/3	1/36	1/36	1/36	1/12
paging	1	0	0	0	0	0	0
I/O-1	1	0	0	0	0	0	0
I/O-2	1	0	0	0	0	0	0
I/O-3	1/2	0	0	0	0	0	1/2
I/O-4	1/2	0	0	0	0	0	1/2
gebruiker	1	0	0	0	0	0	0

Figuur 9.5. Overgangsmatrix usercommando's.

De evenwichtsvergelijking voor de stationaire toestandsvector $\pi = \pi \mathbf{P}$ met voor $\mathbf{P}$ de bovenstaande matrix, blijkt als oplossing te hebben

$$\pi = (36/73, 18/73, 12/73, 1/73, 1/73, 1/73, 4/73)$$

Een usercommando is op de lange duur met een kans van 36 op 73 'bij de CPU'

en met een kans van 4 op 73 'bij de gebruiker'. De CPU krijgt 36 keer de beurt tegen de gebruiker 4, de pagingdrum 18 keer, enzovoort.

Het gemiddeld aantal stappen of beurten tussen twee opeenvolgende keren dat de beurt aan 'gebruiker' is, is volgens de formule uit de vorige paragraaf (§9.8) $1/\pi_{gebruiker} - 1$, dus $1/(4/73) - 1$ of $69/4$.

Bij elk bezoek aan de toestand 'bij de gebruiker' wordt een nieuw usercommando ingebracht. Zo'n usercommando brengt dus in totaal gemiddeld $69/4$ maal een bezoek aan een van de devices CPU, pagingdrum, I/O-1 tot en met I/O-4.

Het gemiddeld aantal malen dat de toestand 'bij de CPU' de beurt heeft tussen twee bezoeken bij de gebruiker in, is $\pi_{CPU}/\pi_{gebruiker}$ of $(36/73)/(4/73) = 9$. Een commando komt dus gemiddeld 9 maal op de CPU. Op dezelfde manier lezen we aan de stationaire toestandsvector af dat een commando gemiddeld $9/2$ maal een page fault genereert (de pagingdrum aandoet), 3 maal een I/O-1 acces pleegt en $1/4$ maal op elk van de devices I/O-2, I/O-3 en I/O-4 komt. In totaal weer $9+9/2+3+3/4 = 69/4$ beurten.

Van de kentallen uit de toestandsvector is duidelijk de relatieve verhouding het belangrijkst, we kunnen stellen dat in (36, 18, 12, 1, 1, 1, 4) de relatieve verhoudingen staan voor de aantallen bezoeken aan de diverse devices. We rekenen van daaruit nog wat verder.

Van de gemiddeld 9 CPU-beurten worden er gemiddeld $1/12 \times 9 = 3/4$ gevolgd door een terugkeer naar de gebruiker. Een commando komt gemiddeld $1/4 + 1/4$ maal bij I/O-3 of I/O-4, daarvan wordt gemiddeld $(1/4+1/4) \times 1/2 = 1/4$ beurt gevolgd door een terugkeer naar de gebruiker. Van de gemiddeld $69/4$ beurten van een commando worden er gemiddeld $3/4+1/4 = 1$ gevolgd door een bezoek aan 'gebruiker'. Dat is natuurlijk geen nieuws, het bewijst alleen dat we consistent redeneren. Terugkeer naar de CPU vanaf de pagingdrum en I/O-1 tot en met I/O-4 vindt plaats in gemiddeld $(9/2+3+1/4+1/2 \times 1/4+1/2 \times 1/4)$ beurten, dit is in 8 beurten. De gemiddeld 9 CPU-beurten bestaan uit gemiddeld deze 8 'terugkeer'-beurten, plus de beurt aan het begin van de verwerking.

Het type rekenwerk dat we hier nu afwerken, kunnen we altijd ook zo rangschikken dat de gemiddelde aantallen beurten worden berekend zonder te verwijzen naar een overgangsmatrix. De redeneringen en het rekenwerk blijken echter dan erg 'error prone' te zijn, het is gemakkelijk een beurt als de beginbeurt bij de CPU te vergeten. Bovendien wordt het voor wat ingewikkelder overgangsmatrices snel heel lastig de juiste sluipweg te vinden en —wat veel erger is— de te maken veronderstellingen worden zeer onduidelijk. Beter is het om altijd vast te houden aan de bepaling volgens $\pi = \pi\mathbf{P}$. Dat is een snelle en zekere weg, die bovendien rechtstreeks kan worden omgezet in een computerprogramma.

Om aan het alternatief ten volle recht te doen, geven we hier voor de liefhebbers de redenering om buiten $\pi = \pi\mathbf{P}$ om de gemiddelde aantallen beurten bij de CPU te vinden. Een commando, waarvan de executie wordt afgebroken of gestaakt, komt met een kans van $1/12$ (rechtstreeks) + $(1/36 \times 1/2 + 1/36 \times 1/2)$ (indirekt) niet weer op de CPU, maar bij de gebruiker. Na elke CPU-beurt volgt dus met

een kans van $1/12+1/36 = 1/9$ een bezoek aan de gebruiker. Volgens de van de binomiale verdeling afgeleide geometrische verdeling (maar die geldt hier eigenlijk helemaal niet, zoals we in §9.8 zagen) duurt het daarom gemiddeld $1/(1/9) = 9$ pogingen voordat het succes van terugkeer naar de gebruiker optreedt. Een commando keert gemiddeld 8 maal terug naar de CPU. Samen met de beginbeurt doet het dus gemiddeld 9 maal de CPU aan.

9.9.2. relatieve bezoekfrequenties en verwerkingsduren

Toepassing van de in het voorbeeld gevonden samenhang tussen overgangskansen en aantallen beurten is vaak de eerste stap in een prestatie-analyse van een computerconfiguratie. We plaatsen de resultaten daarom in een ruimer kader.

In een computerconfiguratie worden opdrachten verwerkt (commando's, jobs, jobsteps, usercommando's, transakties, programma's, enzovoort; we houden ons aan de naamgeving die we al bij de responsetijdrelaties gebruikten). Een randapparaat of CPU noemen we algemeen een device. Een opdracht die door een bepaald device is verwerkt, vertrekt daarop naar een volgend device voor nadere verwerking. Dit volgende device ligt in het algemeen niet vast, nee, er is een bepaalde verdeling over de devices die als volgend device optreden. Deze verdeling wordt de *vertakkingsverhouding* genoemd. De vertakkingsverhouding bij een bepaald device voor een bepaalde soort opdrachten geeft voor deze soort opdrachten de relatieve frequenties aan van de overgangen naar de mogelijke volgende devices.

In het voorbeeld zijn de devices CPU, pagingdrum, I/O-1 tot en met I/O-4 en de gebruiker (ja, hij ook). De soort opdrachten is usercommando's. De vertakkingsverhouding bij de CPU is over de als volgend device optredende (pagingdrum, I/O-1 tot en met I/O-4, gebruiker) als (1/2:1/3:1/36:1/36:1/36:1/12). De vertakkingsverhouding bij I/O-4 over de potentieel als volgend device optredende (CPU, pagingdrum, I/O-1, I/O-2, I/O-3, gebruiker) is als (1/2: 0: 0: 0: 0: 1/2). Analoog voor de andere devices. Het is het handigst de vertakkingsverhouding als kans te normeren door de som van de elementen in de vertakkingsverhouding op 1 te brengen. Het is duidelijk dat steeds de als kans genormeerde vertakkingsverhoudingen samen een overgangsmatrix vormen, die relevant is voor de betrokken soort opdrachten. Voor het voorbeeld staat deze in figuur 9.5.

Door $\pi = \pi \mathbf{P}$ uit te rekenen voor deze overgangsmatrix **P** en te normeren op *som van de kentallen* in $\pi = 1$, vonden we uit de vertakkingsverhoudingen de stationaire evenwichtsvector π met de kansen op de beurt π voor de verschillende devices. Er kwam ook naar voren dat de verhoudingen van de kentallen uit π de relatieve frequenties aangeven, waarmee de devices worden aangedaan: de relatieve *bezoekfrequenties* dus (*visit ratio*). De verhouding $\pi_{CPU}/\pi_{I/O-i}$ geeft aan hoeveel vaker de CPU dan de I/O-*i* de beurt heeft, hoeveel vaker de CPU wordt aangedaan dan de I/O-*i*. Door de evenwichtsvergelijkingen op te lossen vinden we dus algemeen uit de vertakkingsverhoudingen de relatieve bezoekfrequenties.

Het kan voordelen bieden af te stappen van de normering op 'som van de $\pi_i = 1$'. Als we een normering kiezen met $\pi_{gebruiker} = 1$ staan op de overige plaatsen in de

vector 'π' de aantallen keren dat de devices CPU, pagingdrum en I/O-1 tot en met I/O-4 per opdracht worden aangedaan, 'π' is dan geen toestandsvector meer. Dat is verreweg de handigste normering voor de beschrijving van wat er met een opdracht gebeurt. De elementen uit π geven dan per soort opdracht aan hoe vaak gemiddeld beslag wordt gelegd op de devices. Maar het staat natuurlijk vrij te normeren op $\pi_{pagingdrum} = 1$, de overige elementen geven dan aan hoe vaak de overige devices per page fault worden aangedaan. In het voorbeeld worden bijvoorbeeld per page fault 2/9 opdrachten verwerkt en 2/3 maal een acces gepleegd op I/0-1.

Een beurt bij een device heet in het computerjargon een acces. Als we normeren op $\pi_{gebruiker} = 1$ staan in π de gemiddelde aantallen accessen per opdracht bij de verschillende devices!

De gemiddelde aantallen accessen per opdracht (de bezoekfrequenties dus) zullen meestal gebruikt worden om uit te rekenen hoelang de opdrachten gemiddeld door de devices verwerkt worden. We gaan na hoe deze berekening verloopt.

De verwerking van een 'acces' op het device duurt de *serviceduur van het acces* bij het device (dit is niet de duur van het bezoek aan het device: dat zal de som zijn van de serviceduur plus de wachtduur bij het device, anders gezegd: de *verblijfduur* per acces). De totale tijd dat een opdracht door een device wordt verwerkt is de *verwerkingsduur* van de opdracht bij het device. Bij de CPU is de gemiddelde verwerkingsduur van een opdracht de gemiddelde CPU-tijd per opdracht. De gemiddelde verwerkingsduur door een device per opdracht is het produkt van de gemiddelde serviceduur per acces en het gemiddeld aantal accessen per opdracht:

$$\textit{gem. verwerkingsduur per opdracht} = \textit{gem. aantal accessen per opdracht} \times \textit{gem. serviceduur per acces}$$

De gemiddelde serviceduur per acces is vaak voor alle soorten werk gelijk. Meestal is goed bekend hoe groot deze is; het betreft informatie die de fabrikant verstrekt of die door meting verkregen wordt.

De vertakkingsverhoudingen zullen voor de verschillende soorten werk verschillend zijn. Oplossen van de evenwichtsvergelijkingen per soort geeft voor de soorten werk de verhouding van de π's en daarmee de gemiddelde aantallen accessen per opdracht. Daarna volgen voor elke soort werk de gemiddelde verwerkingsduren per opdracht bij de verschillende devices door deze gemiddelde aantallen accessen per opdracht simpelweg te vermenigvuldigen met de gemiddelde serviceduren per acces (zie bijvoorbeeld tabel 9.1). Deze verwerkingsduren van de soorten werk specificeren de werklast!

9.9.3. knelpunten

De gemiddelde verwerkingsduur per opdracht bij een device is basis-invoer bij de modellering van computerconfiguraties, zowel in de knelpunten-analyse als bij separabele queuing-netwerken (§14.9.2). Maar ook daarbuiten is direkt in te zien hoe groot het nut van deze grootheden is. We breiden daarom het voorbeeld uit §9.9.1 tenslotte uit met informatie over de gemiddelde serviceduren per acces; we nemen waarden, die het rekenwerk wat bekorten; de gegevens en de uitkomsten voor de usercommando's (dat zijn hier de 'opdrachten') zijn samengevoegd in tabel 9.1.

1	2	3	4	5
device	accessen per commando	service-duur per acces	verwerkings-duur per commando	door-stroom $\leqslant$
CPU	9	10/9	10	100
paging	9/2	20/9	10	100
I/O-1	3	10/3	10	100
I/O-2	1/4	80	20	50
I/O-3	1/4	80	20	50
I/O-4	1/4	120	30	33.3

Tabel 9.1. Knelpunten bij de verwerking van de usercommando's. Gegevens verwerking per device; tijden in msek., doorstromen in aantallen per sek. Kolom 2-5: gemiddeld aantal accessen per commando, gemiddelde serviceduur per acces, gemiddelde verwerkingsduur per commando en de begrenzing van de doorstroom, die uit deze gegevens volgt. De doorstroom is maximaal 33.3 commando's/sek. Het knelpunt ligt bij I/O-4.

In kolom 2 staan de gemiddelde aantallen accessen per commando (de bezoekfrequenties) uit §9.9.2. Kolom 3 geeft de gemiddelde serviceduren per acces. De gemiddelde serviceduur per acces bij de pagingdrum is 20/9 msek., bij I/O-2 en I/O-3 80 msek. en bij I/O-4 120 msek. Kolom 4 bevat het produkt van het gemiddeld aantal accessen per commando uit kolom 2 en de gemiddelde serviceduur per acces uit kolom 3; dit is de gemiddelde verwerkingsduur per commando. De gemiddelde verwerkingsduur bij de CPU is 10 msek., een commando zal gemiddeld 10 msek. (som van executietijd in user mode en in system mode) door de CPU verwerkt worden.

Laten we eerst de niet-realistische aanname maken dat de usercommando's het rijk alleen hebben, er worden geen devices met andere programmatuur samen gebruikt ('geshared'). Verder gaan we er van uit dat de devices een vaste verwerkingssnelheid hebben. De doorstroom aan usercommando's zal, voor wat I/O-4 betreft, nooit groter kunnen worden dan 1000/30 = 33.3 commando's/sek. Want als de

I/O-4 zich continu bezighoudt met usercommando's werkt hij er in 1000 msek. maar 1000/30 af, of 33.3 per sekonde; ieder commando kost hem immers gemiddeld een verwerkingsduur van 30 msek. De pagingdrum zou volgens dezelfde berekening 100 commando's per sekonde kunnen verwerken. De doorstroom aan usercommando's moet bij elk device natuurlijk gelijk zijn. De pagingdrum komt daarom nooit toe aan wat hij zou kunnen halen, want de I/O-4 begrenst de doorstroom tot onder de 33.3 commando's per sekonde. Volgens dezelfde berekening zouden de I/O-2 en de I/O-3 maximaal 50 commando's per sekonde kunnen verwerken, maar ook zij verwerken er hoogstens 33.3. De CPU en de I/O-1 kunnen net als de pagingdrum maximaal 100 commando's per sekonde verwerken. De maximale doorstroom aan usercommando's is dus 33.3 commando's per sekonde. Als het heel druk is (veel kwiek werkende terminalisten met usercommando's) raakt alleen de I/O-4 vrijwel continu bezet, dit device is *knelpunt* of *bottleneck*. De I/O-4 heeft de langste gemiddelde verwerkingsduur en beperkt daardoor de doorstroom.

Wanneer er wel devices gezamenlijk gebruikt worden door verschillende soorten werk moet de bepaling van het knelpunt worden aangepast (tabel 9.2).

1	2	3	4
device	verwerkings-duur per commando	U ander werk	door-stroom ≤
CPU	10	0.60	40
paging	10	0.08	92
I/O-1	10	0.70	30
I/O-2	20	0	50
I/O-3	20	0	50
I/O-4	30	0	33.3

Tabel 9.2. Knelpunten bij de verwerking van de usercommando's. Er is ook hinder van ander werk. Kolom 3: bezettingsgraad ander werk. Kolom 4: begrenzing doorstroom. Doorstroom maximaal 30 commando's/sek. Knelpunt bij I/O-1.

De CPU, pagingdrum en de I/O-1 zijn respectievelijk 60, 8 en 70% van de tijd bezet met ander werk, bijvoorbeeld systeemwerk of werk van andere interaktieve gebruikers (kolom 3 met de bezettingsgraden U van ander werk). Omdat de CPU 60% van de tijd met iets anders bezig is kan hij in de resterende 40% van zijn tijd per 1000 msek. maar hoogstens 400/10 = 40 commando's verwerken. De grens voor de doorstroom ligt onder deze omstandigheden volgens kolom 4 bij I/O-1; die is dan bottleneck. De maximale doorstroom is 30 commando's per sekonde.

Soms kunnen devices, als ze het druk krijgen, sneller gaan werken (tabel 9.3).

1	2	3	4	5
device	verwerkings-duur per commando	U ander werk	maxi-maal sneller	door-stroom ⩽
CPU	10	0.60	1×	40
paging	10	0.08	1.5×	138
I/O-1	10	0.70	2×	60
I/O-2	20	0	1×	50
I/O-3	20	0	3×	150
I/O-4	30	0	2×	66.6

Tabel 9.3. Knelpunten bij de verwerking van de usercommando's. Devices sneller bij drukte. Kolom 4 geeft aan hoeveel sneller de devices gaan werken als ze het druk krijgen. Kolom 3: bezettingsgraad ander werk. Kolom 5: begrenzing doorstroom. Doorstroom maximaal 40 commando's/sek. Knelpunt bij CPU.

Als I/O-4 hoogstens 2 maal zo snel kan worden (kolom 4), kan hij hoogstens 66.6 commando's per sekonde aan (kolom 5). De grens voor de doorstroom blijkt bij de specificaties uit tabel 9.3 bij de CPU te liggen; die is nu knelpunt. De CPU grenst de doorstroom af tot maximaal 40 commando's per sekonde.

Merk op dat de hardware-snelheden van de devices, in dit geval de omgekeerden van de gemiddelde serviceduren per acces, helemaal niet rechtstreeks uitmaken welk device bottleneck is. In eerste instantie telt niet de serviceduur per acces, die aangeeft hoe snel het device qua hardware is, maar het produkt van deze serviceduur met het gemiddeld aantal accessen per opdracht, dus de verwerkingsduur per opdracht. In de verwerkingsduur is verwerkt hoe vaak van het device gebruik wordt gemaakt. Een qua hardware langzaam device dat weinig gebruikt wordt zal geen knelpunt zijn! Een snel device, dit heeft dus een relatief korte verwerkingsduur, kan toch problemen geven als het ook veel gebruikt wordt door ander werk.

9.10. LUKRAAK WAARNEMEN

We hebben benadrukt dat de stationaire kansen 'kansen op de beurt' zijn. In de voorgaande hoofdstukken zetten we deze *kansen op optreden* naast de *kansen op aantreffen,* de tijdgemiddelde kansen. Er was een groot verschil. De kansen op aantreffen waren kansen volgens de visie van de lukrake waarnemer. Wat zou diens kijk zijn op de schietoefening?

De lukrake waarnemer komt op een lukraak moment aanlopen en neemt bij aankomst lukraak een schutter in het oog. Er is nu informatie nodig over de relatieve duren van de toestanden, die hij bij die schutter kan aantreffen. Als toestanden gaan we onderscheiden: raak gevolgd door raak (*rr*), raak gevolgd door mis (*rm*), mis gevolgd door raak (*mr*) en mis gevolgd door mis (*mm*). Veronderstel dat de duren van deze toestanden —dus de tussenpozen— zich verhouden als 3:2:4:1. De kansen op optreden —de relatieve frequenties— van de toestanden rr, rm, mr en mm zijn in de stationaire fase 4/7, 1/7, 1/7 en 1/7 (§9.5.2). De kans dat de lukrake waarnemer toestand *rr* aantreft is, volgens de basisrelatie voor de kansen op het optreden en aantreffen van duren uit het hoofdstuk BEURTEN EN KANSEN

$$p_{rr} = \frac{4/7 \times 3}{4/7 \times 3 + 1/7 \times (2+4+1)} = 12/19$$

De aantrefkansen voor *rm*, *mr* en *mm* zijn idem 2/19, 4/19 en 1/19 (§7.2.3).

De lukrake waarnemer constateert als hij in toestand *rr* of *mr* binnenkomt, dat er raak wordt geschoten; voor hem is de kans op raak $p_{rr} + p_{mr}$. Hij vindt dus een schutter die raak schiet in 12+4 = 16 van de 19 gevallen (84% raak en niet 5/7 = 71% raak).

toestand	verhouding van de duren	kans op optreden (b, π)	kans op aantreffen (p)	waargenomen
rr	3	4/7	12/19	raak
rm	2	1/7	2/19	mis
mr	4	1/7	4/19	raak
mm	1	1/7	1/19	mis

Tabel 9.4. Kansen op raak en mis voor lukrake waarnemer in standaardvoorbeeld.

Er is een 'subtiel' verschil tussen de lukrake waarnemer en de in §9.6 paraderende commandant en koning. Uit de beschrijving van het gedrag van de geïnteresseerde toeschouwers commandant en koning valt af te lezen dat zij een lukraak schot bekijken. Zij zien dus een lukrake toestandswisseling. De lukrake waarnemer komt daarentegen op een lukraak moment tussen (of bij) twee schoten in. Merk op hoe nodig het is aan te geven wat er precies 'lukraak' wordt gedaan.

De commandant en de koning nemen de kans op optreden waar. Gemiddeld vinden ze dat toestand$_i$ met een kans π_i heersend wordt —zoals we opmerkten.

Uitgespeld naar de toestanden *rr* en *mr* is voor hen de kans op raak niet p_{rr} + p_{mr}, maar

$$b_{rr}+b_{mr} = \pi(0)\times 4/5 + \pi(1)\times 1/2 = 5/7\times 4/5+2/7\times 1/2 = \pi(0) = 5/7$$

Het subtiele verschil met de lukrake waarnemer is weer het bekende verschil uit het hoofdstuk BEURTEN EN KANSEN.

9.10.1. *verdelingen*

Met kansen op optreden corresponderen altijd verdelingen van stochasten, zoals in het hoofdstuk BEURTEN EN KANSEN te zien is. Zo ook in de stationaire fase.

Als de stochast Z de waarde Z_i heeft in de toestand$_i$, is de gemiddelde waarde van de stochast in de stationaire toestand (maar ook gemiddeld over de eerste N stappen, met N groot) (de som loopt over de mogelijke toestanden i)

$$E(Z) = \sum_i Z_i \pi_i$$

Zo'n stochast is bijvoorbeeld de duur van een toestand. Als een schutter 14/5 minuten moet wachten op zijn volgende poging als hij raak heeft geschoten en 5/2 minuten als hij gemist heeft, schiet een schutter met gemiddelde tussenpozen van $14/5\times 5/7+5/2\times 2/7 = 19/7$ minuten. In het hoofdstuk BEURTEN EN KANSEN gingen we al verder in op het gebruik van dit soort overwegingen.

9.11. SAMENVATTING

Markovketens treden op als er eindig veel toestanden zijn, waartussen stapsgewijs overgangen plaats vinden. De overgangsmatrix **P** beschrijft de kansen om daarbij van de ene naar de andere toestand over te gaan. Van groot belang is de stationaire toestand, die optreedt bij ergodische Markovketens. Het stationaire gedrag wordt beschreven door de evenwichtskansen π_i op optreden, op de 'beurt', voor toestand$_i$. Deze kansen zijn de oplossingen van de evenwichtsvergelijkingen $\pi=\pi\mathbf{P}$. In deze evenwichtsvergelijkingen staat hoe men van overgangskansen naar kansen op optreden komt. Het stelsel $\pi=\pi\mathbf{P}$ is afhankelijk, de kentallen uit π zijn op een factor na bepaald; ze beschrijven de relatieve frequenties waarmee de toestanden optreden. Het zijn —bij een normering op kansen— de kansen op de lange termijn en over de lange termijn.

Bij zich stapsgewijs ontwikkelende systemen in de stationaire fase bestaat het rekenwerk, nodig voor het berekenen van de performance, uit het opstellen van **P** en het oplossen van $\pi=\pi\mathbf{P}$.

9.12. OPGAVEN

opgave 9.1: vervolg standaardvoorbeeld (9.2.1)

- Bepaal voor schutters die zojuist gemist hebben de kans om bij het 5-de schot raak te schieten. Bepaal ook de kans om bij het 5-de en bij het 6-de schot raak te schieten. Idem bij het 5-de en bij het 7-de schot. Net zo de kans om bij 1111-de schot te missen. Schrijf de resultaten helemaal met matrices.
- Iemand gaat naar een van de schutteroefeningen kijken. Hij kiest lukraak een schutter, die op het punt staat te gaan schieten. Wat is de kans dat het derde schot van deze schutter raak is? Wat is de kans dat het eerste en het derde schot raak is? En de kans dat het 2-de, 4-de en 6-de schot mis is?

opgave 9.2: studiezin

De studiegewoonten van een student zijn als volgt. Als hij op een werkavond heeft gestudeerd is er 70% kans dat hij de eerstvolgende werkavond niet studeert. Hij vermorst nooit twee werkavonden achterelkaar.

- Hoe groot is de kans dat hij de vijfde werkavond verzuimt, als hij de derde werkavond studeerde?
- De student zal vanavond met een kans van 2 op 3 gaan studeren. Wat zal hij overmorgenavond doen? (vanavond, morgenavond en overmorgenavond zijn werkavonden.)
- Welk percentage van het aantal werkavonden zal hij —over langere tijd bekeken— studeren (59%)?

opgave 9.3: dubbel stochastisch

Een stochastische matrix heeft de eigenschap dat de som van de kentallen in dezelfde rij 1 is. In een dubbel-stochastische matrix geldt deze eigenschap ook voor de kolommen; bij zo'n matrix is de som van de kentallen in een rij 1 en in een kolom 1.

- De 4-bij-4 matrix **P** is dubbel-stochastisch. Bepaal voor deze matrix de limietwaarde Π.

opgave 9.4: verwerkingsduren

Een bepaald type opdrachten uit een configuratie met meerdere processoren en randapparaten maakt uitsluitend gebruik van de processor P1 en de disk R1.

Van deze opdrachten start 2/3 deel met verwerking door P1 en 1/3 deel met verwerking door R1. Na verwerking door P1, wat gemiddeld 30 msek. duurt, volgt in

99 van de 100 gevallen verwerking door R1; na verwerking door R1, wat gemiddeld 20 msek. duurt, volgt in 95 van de 100 gevallen verwerking door P1.

- Hoelang is de verwerkingsduur per opdracht bij P1 en bij R1? Valt er een knelpunt aan te wijzen (wat wordt aangenomen)?

opgave 9.5: memory modules met memory contention

In een computersysteem worden programma's door twee processoren uitgevoerd. Het geheugen van het systeem bestaat uit twee geheugenmodules P en Q. De beide processoren verwerken steeds gedurende een processor-cyclus een instruktie en vragen dan aan een van de geheugenmodules de volgende instruktie af te geven. Als de aanvraag gehonoreerd wordt komt deze instruktie na afloop van een geheugen-cyclus beschikbaar en kan aan de volgende processor-cyclus worden begonnen, enzovoort. Een bepaalde processor is niet gekoppeld aan een bepaalde geheugenmodule; elke processor heeft in principe toegang tot beide modules. Natuurlijk kan een module per geheugen-cyclus slechts één geheugen-acces verwerken.

Als de processoren na afloop van een processor-cyclus beide een acces op dezelfde module wensen (memory contention), moet een van beide gedurende de volgende processor-cyclus wachten. Na de dan volgende geheugen-cyclus worden de gegevens voor de wachtende processor door het geheugenmodule afgeleverd. De beide processoren en de geheugenmodules werken dus strikt onderling synchroon; de tijdsindeling kan aangegeven worden als geheugen-cyclus, processor-cyclus, geheugen-cyclus, enzovoort.

Men houdt voor beide processoren bij in welke tijdsvolgorde de modules gebruikt worden. Op grond daarvan weet men dat elk van de beide processoren na afloop van een processor-cyclus met een kans van 1/2 om een bepaald module vraagt; de accessen zijn lukraak verdeeld over de modules. De toestand waarin de beide processoren zich bevinden wordt steeds vastgesteld na afloop van een processor-cyclus, vlak voor het begin van de geheugen-cyclus. De toestand waarin er i accessen voor module P uitstaan en j voor module Q noemen we de toestand (i,j).

- Welke toestanden zijn mogelijk ((2,0) (1,1) (0,2))? De eerste rij in de overgangsmatrix is (1/2,1/2,0). Bepaal de tweede en de derde rij.

- Thans is het systeem in toestand (1,1). Beide processoren kunnen dus na afloop van de komende geheugen-cyclus aan het werk gaan. Bepaal hoe groot de kans is dat ze dit na de 10-de toestandswisseling ook weer kunnen (1/2). Geef het antwoord ook in formulevorm.

- Men vraagt in welke toestand de processoren zich gemiddeld bevinden na afloop van een geheugen-cyclus. Formuleer het antwoord in woorden, niet alleen formeel (1/4, 1/2, 1/4).

- Uit de gegevens van de fabrikanten blijkt dat een processor-cyclus driemaal zo lang duurt als een geheugen-cyclus. Bepaal gedurende welk deel van de tijd er memory contention optreedt (een van de beide processoren wacht omdat zijn acces niet gehonoreerd wordt).

opgave 9.6: system states

De CPU van een computersysteem is altijd bezig met òf de executie van statements uit een gebruikersprogramma, òf met een routine uit het besturingssysteem (bijvoorbeeld een I/O-opdracht, die voor een userprogramma wordt uitgevoerd), òf met een routine uit het besturingssysteem, die wordt uitgevoerd om een evenwichtige afwikkeling van alle programma's te bevorderen (bijvoorbeeld scheduling); anders zit de CPU in de idle loop. We zullen kortweg zeggen dat de CPU in deze gevallen zich bevindt in de toestanden problem-state, user-supervisor-state, system-supervisor-state en idle-state. Met behulp van in het operatingsysteem ingebouwde tellers (software-monitor) kon worden nagegaan dat

de toestand idle wordt alleen bereikt vanuit de system-supervisor-state; op de toestand idle volgt altijd de toestand system-supervisor-state,
vanuit de problem-state komt de CPU even vaak in user-supervisor-state als in system-supervisor-state,
vanuit de user-supervisor-state komt de CPU driemaal zo vaak in problem-state als in system-supervisor-state,
vanuit de system-supervisor-state komt de CPU in 1 op de 10 gevallen in de idle-state, in alle andere gevallen komt hij in de problem-state.

- Stel de overgangsmatrix voor toestandsveranderingen op. Waarom zijn de diagonaalelementen uit de overgangsmatrix nul? Is dit voor iedere overgangsmatrix het geval?
- De CPU bevindt zich op een bepaald moment in de system-supervisor-state. Volgens de overgangsmatrix is de kans om bij de volgende overgang in de idle-state terecht te komen steeds 1/10, wat ook de vorige toestanden zijn geweest. Is dit redelijk, of is het een benadering? Doe suggesties om mogelijke tekortkomingen te verbeteren.
- Op een bepaald moment is de CPU in de toestand idle. Een poosje later heeft de CPU 3 overgangen gemaakt. Hoe liggen de kansen op de dan optredende toestanden? Op een bepaald moment is onbekend in welke toestand de CPU is, maar het systeem funktioneert al geruime tijd onder normale omstandigheden. Beantwoord ook dan deze vraag (0.03, 0.31, 0.22, 0.44).
- Op een bepaald moment (tijdstip 0) is de CPU in de toestand idle. Hij maakt daarna 1000 overgangen en wordt dan gestopt (tijdstip t). Hoe vaak komt de CPU in die tijd in de toestand idle?

- De tijdsduur tussen het verlaten van de toestand idle en het eerste daarop volgende terugkeren in deze toestand is een aktieve sessie van de CPU. Bepaal hoe vaak de CPU tijdens een aktieve sessie gemiddeld in de user-supervisor-state terecht komt (7.2).

opgave 9.7: vervolg tussenpozen (9.10)

De tussenpozen *rr*, *rm*, *mr* en *mm* zijn precies 3, 2, 4 en 1 minuut.

- Wat is de gemiddelde tussenpoos (vergelijk de berekening in §9.10.1)?
- Een lukrake waarnemer komt op een lukraak moment aanlopen en zoekt aansluitend lukraak een schutter uit om diens schoten te registreren. Hoelang duurt het gemiddeld nog totdat deze schutter schiet? En totdat deze schutter voor het eerst raak schiet?

opgave 9.8: vervolg multiplex verbinding (opgave 8.3)

Het gebruik van de verbinding wordt in de tijd dat Honhio aktief is tijdens de spitsuren met een monitor vastgelegd. Het blijkt dat een periode met twee aktieve locaties tijdens de spitsuren gemiddeld 25 minuten duurt, een periode met drie aktieve locaties duurt gemiddeld 50 minuten en een periode met vier aktieve locaties duurt gemiddeld 5 minuten. Een periode met drie aktieve locaties gaat in 1 op de 6 gevallen over in een periode met twee aktieve locaties en in 5 op de 6 gevallen over in een periode met vier aktieve locaties. Op een periode met twee aktieve locaties en op een periode met vier aktieve locaties volgt altijd een periode met drie aktieve locaties. Het komt (vrijwel) niet voor dat Honhio alleen aktief is of dat er meer dan vier locaties aktief zijn.

De verdeling van de 'tussenpozen' tussen de momenten waarop Honhio de verbinding kan gebruiken is als volgt. Als er naast Honhio nog een andere locatie aktief is, is de tussenpoos gemiddeld 2 sekonden met een kleine spreiding daarom heen. Als er in totaal drie locaties aktief zijn is een tussenpoos in 1 van de 4 gevallen precies 2 sekonden en in 3/4 van de gevallen is de tussenpoos negatief exponentieel verdeeld rond een gemiddelde waarde van 4 sekonden. Als er in totaal vier locaties aktief zijn is de tussenpoos gemiddeld 5 sekonden met een variatiecoëfficiënt van 0.5. Er komt gemiddeld 1 bericht per 3.5 sekonden bij Honhio binnen, berichten komen op lukrake momenten. Het transport van een bericht duurt gemiddeld 1 msek.

- Geef de overgangsmatrix tussen de toestanden met twee, drie en vier aktieve locaties. In welk deel van de perioden zijn er drie aktieve locaties (1/2)?
- De beheerder van Honhio gaat tijdens de spitsuren af en toe zo maar eens kijken hoeveel locaties er aktief zijn. In welk deel van de gevallen vindt hij dat er drie locaties aktief zijn (6/7)?
- Bepaal tenslotte hoelang het gemiddeld duurt voordat een bericht dat tijdens de spitsuren bij Honhio binnenkomt, is overgestuurd (3.36).

10

Eigenschappen Markovketens

10.1. INLEIDING

In het vorige hoofdstuk leerden we de ergodische eerste orde Markovketens kennen. We willen verder ingaan op wat er bijzonder is aan deze ketens.

Eerst worden Markovketens heel in het algemeen bekeken. Markovketens blijken altijd uiteen te vallen in de basisvormen *absorberend*, *periodiek* en *ergodisch*. In Markovketens met een eindig aantal toestanden komt elk systeem, als er geen periodiek gedrag is, op den duur in een toestand uit een 'ergodische' groep toestanden. De overgangskansen voor de toestanden uit deze groep vormen een *ergodische* overgangsmatrix. Voor het doorrekenen van de aanloop naar zulke groepen kan een ergodische groep toestanden worden vervangen door een enkele *absorberende toestand*, waaruit geen overgangen naar andere toestanden mogelijk zijn.

De stationaire toestand bij ergodische Markovketens blijkt te ontstaan door een voortschrijdende middeling. We leren voor hogere-orde Markovketens de overgangsmatrix van de eerste orde op te stellen en gaan in op het verband tussen een overstappatroon en de overgangsmatrix.

10.2. ALGEMENE MARKOVKETENS

10.2.1. indeling

Bij eerste orde Markovketens hangen de overgangskansen alleen van de huidige toestand af, deze overgangskansen staan in de stochastische overgangsmatrix (§9.3). Niet alle stochastische matrices hebben de eigenschappen, die we in het vorige hoofdstuk toedichtten aan 'ergodische' overgangsmatrices. Bij Markovketens kunnen in het algemeen in de op elkaar volgende toestandsovergangen drie

verschillende patronen worden onderscheiden. Deze specifieke patronen zijn in de praktijk snel te overzien. Er moet worden gelet op:

- *[periodiek gedrag]*
  Soms herhaalt een situatie zich systematisch na een zeker aantal stappen. Er treden vaste 'lussen' op in de op elkaar volgende toestanden. Gevallen met dergelijke periodieke aspekten blijken in concreto nooit onhanteerbaar anders te zijn; een remedie is vaak de periodiciteit apart te zetten en apart te behandelen.
- *[keten reducibel]*
  de toestanden vallen uiteen in twee of meer groepen, die niets met elkaar te maken hebben, want er zijn geen overgangen tussen een toestand uit de ene groep en een toestand uit een andere groep. De remedie is hier simpel: houdt voor elke groep een aparte beschouwing. De overgangsmatrix wordt gesplitst in aparte, volledig gescheiden overgangsmatrices.
- *[absorberende groepen]*
  de toestanden vallen uiteen in twee of meer groepen, die zich onderling asymmetrisch gedragen. Er kunnen wel overgangen plaatsvinden van toestanden uit een niet-absorberende groep naar toestanden uit een absorberende groep, maar omgekeerd niet: er zijn geen overgangen mogelijk tussen toestanden uit een absorberende groep naar toestanden uit een niet-absorberende groep. Er is, om zo te zeggen, wel een weg heen naar een absorberende groep, maar geen weg terug.

We gaan niet in op wat er voorspeld kan worden als er periodiciteit optreedt. Maar of er absorberende (groepen) toestanden zijn maakt een essentieel verschil uit. Daarop moeten we greep krijgen. We veroorloven ons daarom een uitstapje naar systemen met absorberende groepen.

10.2.2. doorgangstoestanden (intermezzo Markovketens)

Een illustratief voorbeeld van een algemene (eerste orde) Markovketen zonder periodieke toestanden staat in figuur 10.1. De keten heeft 10 verschillende toestanden, genummerd naar het rijnummer. De toestanden vallen uiteen in verschillende groepen, vandaar de scheidslijnen in de matrix.

Toestand$_1$ is een geïsoleerde toestand, die geen enkel verband heeft met de andere toestanden. De keten is hierdoor reducibel, toestand$_1$ en de rest van de toestanden zijn apart te behandelen; we besteden verder geen aandacht aan toestand$_1$.

Vanuit de groep toestanden (7, 8, 9, 10) zijn overgangen mogelijk naar toestanden uit de groepen (2, 3) en (4, 5, 6). Vanuit een toestand in de groep (4, 5, 6) kan alleen een overgang plaatsvinden naar een toestand uit deze zelfde groep, overgangen naar elders zijn niet mogelijk. Dit is ook zo voor de groep toestanden (2, 3).

De samenhang komt in de overgangsmatrix duidelijk naar voren door de handige nummering van de toestanden. Direkt na de kennismaking met een

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	1/4	3/4	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0
4	0	0	0	1/3	2/3	0	0	0	0	0
5	0	0	0	0	1/4	3/4	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	1/6	1/3	1/2	0	0
9	0	0	1/4	0	0	0	1/4	0	1/4	1/4
10	0	0	0	1/2	0	0	0	1/6	0	1/3

Figuur 10.1. Een algemene overgangsmatrix.

onbekende overgangsmatrix is het zaak de vrijheid om toestanden te hernummeren uit te buiten; dat is hier gebeurd.

De groep toestanden (4, 5, 6) is een *absorberende groep*: er kunnen wel overgangen plaatsvinden van toestanden buiten deze groep naar de toestanden uit deze groep (vanuit toestand_8 naar toestand_6 en vanuit toestand_{10} naar toestand_4), maar er zijn geen overgangen van de toestanden uit de groep naar toestanden buiten de groep (er zijn geen overgangen vanuit toestand_4, toestand_5 of toestand_6 naar andere toestanden dan deze drie). De overgangen tussen de toestanden uit de groep onderling worden beschreven met de overgangsmatrix

$$\mathbf{P} = \begin{bmatrix} 1/3 & 2/3 & 0 \\ 0 & 1/4 & 3/4 \\ 1 & 0 & 0 \end{bmatrix}$$

Dit is op zich een stochastische matrix.

Ook de groep toestanden (2, 3) is een absorberende groep. De onderlinge overgangsmatrix is

$$\mathbf{P} = \begin{bmatrix} 1/4 & 3/4 \\ 1 & 0 \end{bmatrix}$$

Omdat er helemaal geen overgangen zijn vanuit een absorberende groep naar 'elders', heten absorberende groepen ook wel *gesloten* of *closed* groepen; hier zijn (2, 3) en (4, 5, 6) gesloten groepen.

De toestanden buiten de absorberende groepen zijn *doorgangstoestanden* of *transient states*, ze vormen een *doorgangsgroep*. Bij de doorgangstoestanden (7, 8, 9, 10) liggen de mogelijkheden om overgangen te maken heel anders dan bij

de toestanden uit de absorberende groepen (4, 5, 6) en (2, 3). Er kunnen vanuit de toestanden (7, 8, 9, 10) wel overgangen naar 'elders' worden gemaakt en daardoor zal een systeem dat in een van de toestanden (7, 8, 9, 10) is, ooit andere toestanden dan 7, 8, 9 of 10 gaan bezetten. Zo'n systeem kan bij de volgende overgangen heel lang binnen deze groep toestanden van de een naar de ander gaan. Maar omdat er in de toestanden 8, 9 en 10 een eindige kans is dat de 'fatale' overstap naar de absorberende groepen toestanden wordt gemaakt, zal een systeem het nooit 'tot in het oneindige' uithouden binnen de groep toestanden 7, 8, 9, 10. Het zal altijd, na een aantal malen een van deze toestanden te hebben bezet, òf in een toestand uit de absorberende groep (2, 3) òf in een uit de absorberende groep (4, 5, 6) terecht komen. Het systeem blijft daarna binnen deze absorberende groep. Het verblijf binnen een doorgangsgroep is dus altijd van voorbijgaande aard, vandaar de naam transient state.

De toestanden van een Markovketen vallen steeds uiteen in toestanden binnen doorgangsgroepen en toestanden binnen absorberende groepen. Elke absorberende groep kan op zich weer uiteenvallen in doorgangs(sub)groepen en absorberende (sub)groepen en die weer, en die weer: een recurrente opsplitsing. Uiteindelijk resteert bij elke Markovketen een aantal groepen die niet verder uiteenvallen (we zien af van periodiciteit). Deze groepen heten *irreducibel.* Een gesloten groep toestanden is irreducibel als alle toestanden in deze groep vanuit elk van de toestanden in de groep kunnen worden bereikt binnen een eindig aantal overgangen. Er zijn dus binnen een irreducibele groep geen absorberende (sub)groepen meer.

Via de recurrente opsplitsing wordt elke absorberende groep toestanden uiteengerafeld in toestanden binnen doorgangsgroepen en binnen irreducibele groepen. Een irreducibele gesloten groep toestanden wordt met de fraai-mystieke naam *ergodische groep* aangeduid, die we voortaan aanhouden. De overgangen binnen zo'n groep vinden plaats volgens een ergodische Markovketen. Zulke ketens bespraken we in het voorgaande hoofdstuk.

Er moet voor het verloop in de tijd dus slechts onderscheid worden gemaakt tussen toestanden uit doorgangsgroepen en toestanden uit ergodische groepen. De theorie leert dat dit inderdaad de enige mogelijkheden zijn: toestanden horen òf tot een doorgangsgroep òf het zijn toestanden uit een ergodische groep. De stelling geldt als er geen periodiciteit optreedt en het aantal verschillende toestanden eindig is. De matrices mogen natuurlijk ook niet per stap variëren (§9.8; **P** geen funktie van het stapnummer); die veronderstelling noemt men *tijdhomogeen*. Kortom: voor algemene tijdhomogene niet-periodieke Markovketens met een eindig aantal toestanden horen de toestanden altijd òf tot een doorgangsgroep òf tot een ergodische groep.

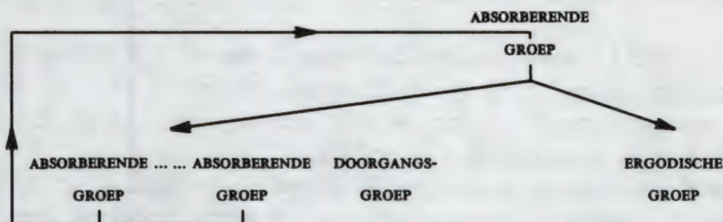
In dit voorbeeld is de groep (7, 8, 9, 10) een doorgangsgroep. De beide groepen toestanden (2, 3) en (4, 5, 6) zijn niet alleen absorberende groepen, maar ook ergodische groepen. De aangegeven overgangsmatrices voor deze groepen zijn *ergodische* overgangsmatrices. Als de overgangsmatrix in de rij die hoort bij toestand_6 een 1 in kolom 6 in plaats van in kolom 4 zou hebben, zou de

absorberende groep (4, 5, 6) niet ergodisch zijn, want de groep valt dan uiteen in de toestand$_6$ (dit is een enkele 'absorberende toestand' volgens de benaming uit §10.2.3) en de doorgangstoestanden toestand$_4$ en toestand$_5$. De overgangsmatrix voor deze groep zou na herordening met als nieuwe volgorde 6,5,4 zijn

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 3/4 & 1/4 & 0 \\ 0 & 2/3 & 1/3 \end{bmatrix}$$

Uit deze vorm blijkt duidelijk dat nu de absorberende groep (6, 5, 4) verder uiteen valt, de twee doorgangstoestanden 5 en 4 springen er uit. De overgangsmatrix voor een absorberende groep bevat voor elke ergodische subgroep altijd een ergodische (sub)overgangsmatrix. De eendimensionale matrix, bestaande uit de 1 linksboven, is hier de supersimpele ergodische overgangsmatrix voor de ergodische groep die alleen toestand$_6$ bevat.

Een schema voor de indeling van de toestanden uit een algemene Markovketen staat in figuur 10.2. Een absorberende groep toestanden die niet ergodisch is, valt altijd uiteen in één of meer absorberende groepen en een groep doorgangstoestanden. Ieder van deze absorberende groepen kan weer op dezelfde wijze worden gesplitst. Uiteindelijk wordt een toestand ingedeeld als toestand uit een doorgangsgroep of uit een ergodische groep.



Figuur 10.2. Indeling toestanden (eindig veel toestanden, niet periodiek). Een absorberende groep toestanden is ergodisch of kan worden opgesplitst in een subgroep doorgangstoestanden en een of meer absorberende subgroepen. Iedere absorberende subgroep valt op deze wijze uiteen totdat elke toestand is ingedeeld bij een doorgangsgroep of bij een ergodische groep.

In het voorbeeld is de groep toestanden (2-10) de absorberende groep op het hoogste niveau (toestand 1 staat al apart). Deze valt uiteen in de absorberende groep (2, 3), de absorberende groep (4-8) en de doorgangsgroep (9, 10).

De absorberende groep (4-8) valt uiteen in de doorgangsgroep (7, 8) en de absorberende groep (4, 5, 6), (de doorgangsgroep (7, 8, 9, 10) wordt volgens dit schema opgesplitst in de groepen (7, 8) en (9, 10)).

De absorberende groepen (2, 3) en (4, 5, 6) vallen niet verder uiteen: het zijn ergodische groepen.

Het is nu duidelijk dat we in het voorgaande hoofdstuk inderdaad niet de meest algemene Markovketens bekeken hebben. Het ging steeds over de belangrijke klasse van de ergodische Markovketens, over de overgangen binnen een ergodische groep. Binnen een ergodische groep worden de kansen op de lange termijn bepaald volgens de evenwichtsvergelijking voor de stationaire toestand, zoals we daar zagen.

10.2.3. op de lange termijn bij algemene Markovketens

Hoe wordt de voorspelling voor de 'kansen op de lange termijn' bij algemene Markovketens?

Het systeem begint misschien in een toestand uit de groep doorgangstoestanden. Na een groter of kleiner aantal overgangen verdwijnt het systeem uit deze groep. Het komt als in een 'fuik' in een van de absorberende groepen terecht, binnen deze absorberende groep spelen zich daarna de overgangen af. Als de absorberende groep geen ergodische groep is, spelen zich op de lange duur de overgangen af binnen een van de absorberende groepen binnen deze absorberende groep —en zo gaat het verder. Uiteindelijk spelen de overgangen zich af binnen een ergodische groep. Als die ergodische groep uit een enkele toestand bestaat wordt deze toestand betiteld als *'absorberende toestand'*. Deze ingeburgerde naam is eigenlijk wat verwarrend, iedere toestand uit een absorberende groep zou met evenveel recht een absorberende toestand kunnen worden genoemd, maar dat wordt niet gedaan. Als het systeem in een absorberende toestand is, blijft het in deze toestand. De overgangen vanuit eèn absorberende toestand zijn altijd weer terug naar deze toestand!

In ons voorbeeld komt elk systeem dat begint binnen (7, 8, 9, 10) op den duur òf in de ergodische groep (2, 3) òf in de ergodische groep (4, 5, 6); bij de iets gewijzigde overgangsmatrix, waarin (4, 5, 6) niet ergodisch is, in de ergodische groep (2, 3) of in de absorberende toestand (6).

De toestand op de lange termijn wordt kennelijk de toestand op de lange termijn binnen de ergodische groep, waarin het systeem uiteindelijk terecht komt. Die toestand wordt beschreven door de stationaire toestandsvector voor die groep. Deze wordt bepaald uit $\pi = \pi\mathbf{P}$, waarin $\mathbf{P}$ de deelmatrix is uit de overgangsmatrix, die de overgangen binnen deze groep beschrijft. In het voorbeeld zijn dat de voor (2, 3) en (4, 5, 6) aangegeven matrices.

De vraag blijft hoe bij algemene Markovketens de kansen liggen om uitgaande van de begintoestand in een bepaalde ergodische groep terecht te komen (te worden 'ingevangen'). Voor de analyse van wat zich over de lange termijn afspeelt is het alleen nodig Markovketens met absorberende toestanden nader te bekijken, het onderzoeken van Markovketens met absorberende groepen toestanden hoeft daarvoor niet. Of anders gezegd: het is voldoende zich te beperken tot systemen met absorberende groepen toestanden, die uit slechts één enkele toestand bestaan.

De argumentatie om dat in te zien is als volgt.

Zodra een systeem een toestand uit een ergodische groep heeft bereikt, wordt zijn lange-termijn gedrag beschreven door de stationaire toestandsvector voor die groep (§9.4). Het gedrag binnen een ergodische groep is op den duur onafhankelijk van de begintoestand. Dat wil hier zeggen onafhankelijk van de toestand uit de ergodische groep, waarlangs de ergodische groep het eerst wordt betreden. We kunnen er rustig van af zien deze detailinformatie bij te houden. Maar dan kunnen we voor ons doel best de hele ergodische groep samenballen tot een geheel, tot één enkele *samengestelde* absorberende toestand. Voor de verdere analyse is slechts de aanloopfase naar de ergodische groep belangrijk, en dat is de aanloopfase naar deze samengestelde toestand. Laten we daarom rustig de hele ergodische groep maar samennemen tot een enkele 'absorberende toestand'!

In het voorbeeld is het niet interessant of de ergodische groep (4, 5, 6) langs $toestand_4$ (vanuit $toestand_{10}$) of langs $toestand_6$ (vanuit $toestand_8$) wordt bereikt. Voor een beschouwing over de kansen op de lange termijn kan worden volstaan met een gereduceerde vorm van de overgangsmatrix (we laten de losse $toestand_1$ helemaal weg):

$$\mathbf{P}_{gereduceerd} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1/6 & 1/3 & 1/2 & 0 & 0 \\ 1/4 & 0 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 1/2 & 0 & 1/6 & 0 & 1/3 \end{bmatrix}$$

Hierin is elke ergodische groep vervangen door een enkele absorberende toestand. Bij de eerste rij hoort de absorberende groep (2, 3), bij de tweede rij de absorberende groep (4, 5, 6) en bij de rijen 3 tot en met 6 horen de doorgangstoestanden 7, 8, 9 en 10.

Een overgangsmatrix wordt door het samentrekken van ergodische groepen tot samengestelde absorberende toestanden tot op zijn skelet gereduceerd. Alleen dat skelet is van belang voor de kansen op de lange termijn. Het doet er niet toe of de absorberende toestanden daaruit samengesteld zijn uit een ergodische groep toestanden of uit een enkele toestand bestaan.

In de theorie van de absorberende Markovketens komen daarom slechts overgangsmatrices met een aantal doorgangstoestanden en één of meer absorberende toestanden aan de orde. Die theorie gaat dan in op de resterende vragen, zoals met welke kans het systeem in een van de onderscheiden absorberende toestanden (eigenlijk dus ergodische groepen) terecht komt, hoelang dat duurt, enzovoort. Hoe de zaken binnen een samengestelde 'absorberende toestand' liggen kan altijd worden nagegaan door deze toestand weer uiteen te rafelen en daarop de analyse van ergodische Markovketens toe te passen, die we hebben bekeken in het voorgaande hoofdstuk.

We sluiten hier ons uitstapje naar de algemene Markovketens af. Absorberende ketens zijn dus nuttig om 'transient' gedrag te beschrijven, dat wil zeggen waar kom je terecht en hoelang duurt het voordat je daar bent? Zo komen usercommando's die rondpendelen in een computerconfiguratie uiteindelijk bij de absorberende toestand 'terminals' terecht, ze zijn dan weer terug bij de gebruikers die hen hebben ingebracht. Boodschappen die in een computernetwerk gerouteerd worden via een adaptief routing algoritme komen uiteindelijk op de plaats van bestemming (absorberende toestand) of bij 'error' (absorberende toestand). Het gaat niet om het gedrag op de lange duur maar over de weg ergens heen. Als een systeem zich stapsgewijs ontwikkelt zijn absorberende Markovketens het middel bij uitstek om dat type gedrag te beschrijven, zoals —we zagen het in het vorige hoofdstuk— ergodische ketens het stationaire gedrag beschrijven. De kans om in een bepaalde ergodische groep (dat wil zeggen in een zekere absorberende toestand) terecht te komen en hoeveel stappen dat gemiddeld duurt, wordt berekend door gebruik te maken van matrixrekening en recurrente betrekkingen, analoog aan de methoden uit §9.5.

Omdat we meer in het stationaire gedrag dan in het aanloopgedrag zijn geïnteresseerd gaan we niet verder in op de theorie van algemene Markovketens, respectievelijk absorberende Markovketens, maar verwijzen naar de leerboeken. De Markovketens, waarvoor we in het vorige hoofdstuk de eigenschappen van $\mathbf{P}^n$ voor grote waarden van n berekend hebben, zijn dus inderdaad de ergodische Markovketens. Er treden daar geen absorberende groepen toestanden op en geen van de aangegeven overige dwarsliggers geeft problemen. Voortaan houden we ons alleen maar bezig met zulke Markovketens.

10.3. STATIONAIR DOOR UITMIDDELEN

Wat is nu de eigenlijke achtergrond, de eigenlijke oorzaak van het stationair worden van de kansen bij ergodische Markovketens? Gelukkig kunnen we dit laten zien. We willen geen strenge bewijzen gaan geven, het volgende is bedoeld als een illustratie van wat er in wezen aan de hand is.

Natuurlijk nemen we weer ons standaardvoorbeeld onder handen. Pas de overgangsmatrix $\mathbf{P}$ toe op een willekeurige kolomvector $\mathbf{a}$, bijvoorbeeld een kolomvector met de kentallen 20 en 60. Dan is

$$\mathbf{Pa} = \begin{bmatrix} 4/5 & 1/5 \\ 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 20 \\ 60 \end{bmatrix} = \begin{bmatrix} 28 \\ 40 \end{bmatrix}$$

De elementen uit $\mathbf{Pa}$ ontstaan uit $\mathbf{a}$ door de elementen uit $\mathbf{a}$ te *middelen*; dit komt omdat $\mathbf{P}$ een stochastische matrix is. Eerst wordt gemiddeld met de gewichten 4/5 en 1/5, daarna met de gewichten 1/2 en 1/2.

Het zal velen uit hun levenservaring bekend zijn, dat het resultaat van het middelen van een aantal getallen iets oplevert dat tussen het grootste en het kleinste getal in ligt. Als er steeds sprake is van echte middeling bij het bepalen van $\mathbf{Pa}$

—dit is bijvoorbeeld het geval als geen enkel gewicht (geen enkel element uit **P**) nul is— zal zowel het resultaat van de eerste middeling als van de tweede middeling tussen het grootste en het kleinste getal uit **a** inliggen. Men noemt vaak het verschil tussen het maximum en het minimum van een verzameling getallen de range van deze getallen. Toepassen van de overgangsmatrix **P** op een kolomvector geeft dus een nieuwe kolomvector met een kleinere range. Hier is de range van de kolomvector na een keer vermenigvuldigen met **P** gekrompen van $60 - 20 = 40$ tot $40 - 28 = 12$.

Het punt is nu: herhaaldelijk toepassen van **P** levert kolomvectoren op met een steeds kleinere range. Bij steeds maar vermenigvuldigen met **P** wordt op den duur de range (vrijwel) nul en zijn de getallen uit de kolomvector $\mathbf{P}^n\mathbf{a}$ alle (vrijwel) gelijk. De limietwaarde van $\mathbf{P}^n\mathbf{a}$ bestaat dus uit een kolomvector met gelijke getallen, zeg allemaal α.

Als we in plaats van **a** een andere kolomvector **b** nemen, vinden we ook een ander getal voor α, zeg β. Een willekeurige matrix kunnen we opgebouwd denken uit een aantal kolomvectoren achter elkaar. Voor elk van deze kolomvectoren geldt de voorgaande redenering. Er zal voor elke matrix **A** gelden dat

$$\lim_{n\to\infty} \mathbf{P}^n\mathbf{A} = \Omega$$

De limietwaarde van $\mathbf{P}^n\mathbf{A}$ is dus de matrix Ω. Voor de twee-bij-twee matrix **A**, opgebouwd uit **a** en **b**, is

$$\Omega = \begin{bmatrix} \alpha & \beta \\ \alpha & \beta \end{bmatrix}$$

Nemen we een **A** die een inverse heeft, dan vinden we na toepassing van $\mathbf{A}^{-1}$

$$\lim_{n\to\infty} \mathbf{P}^n = \Omega\mathbf{A}^{-1}$$

Hier blijkt dat de limiet van $\mathbf{P}^n$ inderdaad bestaat, mits het uitmiddelen bij alle rijen van **P** funktioneert. Bij ergodische ketens is dat —we bewijzen het niet— steeds het geval. Als er absorberende toestanden zijn, of wanneer er periodiciteit is, werkt het uitmiddelen natuurlijk niet.

De limietwaarde $\Omega\mathbf{A}^{-1}$ is een uitdrukking voor de matrix Π, de overgangsmatrix naar de stationaire toestanden. De matrix Ω heeft gelijke rijen en daardoor heeft ook de matrix $\Omega\mathbf{A}^{-1}$ voor iedere $\mathbf{A}^{-1}$ gelijke rijen. Het belang van deze eigenschap voor Π hebben we in §9.5 onderstreept; het maakt dat de begintoestand van geen invloed is op de stationaire toestand.

We hebben nu geleerd dat het constant worden van de toestandsvector gezien kan worden als het resultaat van een voortschrijdende middeling.

Wanneer voor **A** een handige waarde wordt genomen, zal er nog meer te bereiken zijn. Liefst nemen we een **A** waarvoor Ω eenvoudig is aan te geven. Als $\mathbf{A} = \mathbf{P}-\mathbf{I}$ wordt genomen, is Ω gelijk aan de nulmatrix, immers $\mathbf{P}^n(\mathbf{P}-\mathbf{I}) = \mathbf{P}^{n+1}-\mathbf{P}^n \approx 0$. De matrix $\mathbf{A} = \mathbf{P}-\mathbf{I}$ heeft geen inverse, want $\pi = \pi\mathbf{P}$ is een afhankelijk stelsel vergelijkingen.

We maken daarom ook nog gebruik van de matrix **E**, een matrix die evenveel rijen en kolommen heeft als **P** en waarvan alle elementen, ook de elementen buiten de diagonaal, 1 zijn. Als **P** een k-bij-k matrix is, bestaat **E** uit k kolomvectoren $\mathbf{I}(k)$ achter elkaar ((9.1) uit §9.3). De eigenschap dat de som van de elementen in een rij van een overgangsmatrix steeds 1 is, maakt dat voor elke overgangsmatrix **P** geldt

$$\mathbf{PE} = \mathbf{E}$$

Als we nu voor **A** nemen $\mathbf{A} = \mathbf{P} - \mathbf{I} + \mathbf{E}$ wordt

$$\Omega \approx \mathbf{P}^n\mathbf{A} = \mathbf{P}^n(\mathbf{P}-\mathbf{I}) + \mathbf{P}^n\mathbf{E} \approx 0 + \mathbf{E} = \mathbf{E}$$

zodat $\Omega = \mathbf{E}$ en tenslotte

$$\Pi = \mathbf{E}(\mathbf{P}-\mathbf{I}+\mathbf{E})^{-1}$$

We hebben daarmee een handige concrete uitdrukking gevonden voor Π, de overgangsmatrix naar de stationaire fase. Bij een gegeven **P** kan deze matrix Π worden berekend via het inverteren van matrices: bereken $\mathbf{P}-\mathbf{I}+\mathbf{E}$, inverteer deze matrix en voorvermenigvuldig met **E**.

Het standaard alternatief, dat meestal aantrekkelijker is, is natuurlijk om een programma voor het oplossen van vergelijkingen te benutten. Maar het is plezierig om voor elke **P** een uitdrukking voor Π, en daarmee voor π, te hebben.

10.3.1. snelheid van naderen limiet (mathematisch intermezzo)

We kunnen het probleem van het bepalen van de limietwaarde ook op een heel andere manier aanpakken. Veronderstel dat de begintoestandsvector gelijk zou zijn aan de stationaire toestandsvector $(\pi(0), \pi(1))$. De stationaire toestandsvector vonden we als oplossing van

$$(\pi(0), \pi(1))\,\mathbf{P} = (\pi(0), \pi(1))$$

Er geldt dus altijd

$$(\pi(0), \pi(1))\,\mathbf{P}^n = (\pi(0), \pi(1))$$

De stationaire toestandsvector verandert immers niet bij vermenigvuldigen met **P**. Bij de begintoestandsvector $(p_0(0),\ p_0(1)) = (\pi(0),\ \pi(1))$ is het berekenen van $(p_0(0),\ p_0(1))\mathbf{P}^n$ heel erg eenvoudig. Dit is meer in het algemeen het geval voor een rijvector $(u(0), u(1))$ die voldoet aan (met λ geven we een getal aan, dat nog moet worden bepaald).

$$(u(0), u(1))\,\mathbf{P} = \lambda\,(u(0), u(1))$$

waarbij geldt

$$(u(0), u(1))\,\mathbf{P}^n = \lambda^n\,(u(0), u(1))$$

We gaan kijken of we zo'n vector kunnen vinden en wat λ dan moet zijn. We schrijven de vergelijking voor $\mathbf{u} = (u(0), u(1))$ als

$$(u(0), u(1))\,(\mathbf{P}-\lambda\mathbf{I}) = (0, 0)$$

Het bepalen van waarden voor λ, die voldoen aan

$$\mathbf{u}(\mathbf{P}-\lambda\mathbf{I}) = 0$$

is het uit de matrixtheorie bekende probleem van het vinden van de *eigenwaarden* van de matrix **P**. De oplossingen **u** zijn daarbij de *eigenvectoren*. Alleen voor heel bepaalde waarden van λ —de *eigenwaarden*— vormen de vergelijkingen $\mathbf{u}(\mathbf{P}-\lambda\mathbf{I}) = 0$ een afhankelijk stelsel en is er een andere oplossing mogelijk dan het triviale $\mathbf{u} = 0$.

In het standaardvoorbeeld blijkt zonder veel moeite dat $\lambda = 1$, met eigenvector evenredig met $(5, 2)$, en $\lambda = 3/10$, met eigenvector evenredig met $(1, -1)$, de enige mogelijkheden zijn. De stationaire toestandsvector $(5/7, 2/7)$ is dus niets anders dan de correct genormeerde $(u(0) + u(1) = 1)$ eigenvector bij de eigenwaarde $\lambda = 1$.

Eigenvectoren spannen volgens de matrixtheorie een ruimte op, waarin matrix-manipulaties een eenvoudige vorm krijgen, zo ook hier. Ontwikkel dus naar de basisvectoren $\pi = (\pi(0),\ \pi(1)) = (5/7, 2/7)$ en $\mathbf{v} = (1, -1)$. Nu is het resultaat van vermenigvuldigingen met **P** eenvoudig te vinden:

$$\begin{aligned}(p_n(0), p_n(1)) &= (p_0(0), p_0(1))\mathbf{P}^n \\ &= (\pi(0), \pi(1))\mathbf{P}^n + (p_0(0) - \pi(0))(1, -1)\mathbf{P}^n \\ &= \boldsymbol{\pi} + (p_0(0) - \pi(0))(0.3)^n\mathbf{v} \\ &= (5/7, 2/7) + (p_0(0) - 5/7)(0.3)^n(1, -1)\end{aligned}$$

Hier staat dat het gedeelte van de toestandsvector, dat van de stationaire toestandsvector $(\pi(0), \pi(1))$ afwijkt, voor grote n steeds geringer wordt, $(0.3)^n$ wordt immers alsmaar kleiner.

Belangrijk is de algemene weg, die hier te zien is. Om na te gaan hoe snel een toestandsvector tot de stationaire toestandsvector nadert is de route: volg een ontwikkeling naar eigenwaarden. In ons voorbeeld constateren we zo, dat het verschil tussen $(p_n(0), p_n(1))$ en $(\pi(0), \pi(1))$ de n-de term is uit een meetkundige reeks met reden 3/10.

Nu we een hanteerbare uitdrukking voor $\mathbf{P}^n$ hebben kunnen we de limietovergangen in de paragraaf OVER/OP DE LANGE TERMIJN (§9.6.1) uit het vorige hoofdstuk beter uitschrijven, maar we zullen het hier niet doen.

10.4. MARKOVKETENS VAN HOGERE ORDE

10.4.1. hogere orde gelijkwaardig eerste orde

Wanneer bij een Markovketen de kans op de volgende toestand niet alleen afhangt van de huidige toestand, maar ook van de vorige, de daaraan voorafgaande, enzovoort, tot k stappen terug in de tijd, noemden we de Markovketen een Markovketen van de k-de orde (vergelijk §9.3). Bij een Markovketen van de tweede orde bijvoorbeeld is niet alleen de huidige toestand, maar ook de voorgaande toestand bepalend voor de kans dat een bepaalde toestand op de huidige zal volgen.

Het blijkt dat zo'n k-de orde Markovketen ook altijd te beschrijven is als een eerste orde Markovketen, als we maar voor lief nemen dat daarvoor een uitgebreider arsenaal toestanden, en dus een veel grotere overgangsmatrix nodig is.

We nemen als voorbeeld een keten van de tweede orde. De toestand na de n-de stap wordt aangegeven met toestand_n. De mogelijke toestanden worden gemakshalve onderscheiden door letters. De kans om uitgaande van toestand i, bij voorgaande toestand h, na de volgende stap in j terecht te komen (h-i-j als sequentie) is de voorwaardelijke kans

$$Pr(\text{toestand}_{n+1} = j \mid \text{toestand}_n = i, \text{toestand}_{n-1} = h)$$

Bij een eerste orde keten hebben we met de gebruikelijke overgangsmatrix te maken, die geschreven kan worden als (want een overgangsmatrix bevat voorwaardelijke kansen):

$$\mathbf{P}_{ij} = Pr(\text{toestand}_{n+1} = j \mid \text{toestand}_n = i)$$

De eenvoudige truc om ook bij hogere orde ketens deze structuur terug te krijgen bestaat uit het samenvoegen van de sequentie van twee toestanden tot een enkele nieuwe *samengestelde* toestand. Deze toestand —zeg Toestand— wordt gekarakteriseerd door een in tijdvolgorde geordend toestandenpaar volgens de oude notatie. Dus $\text{toestand}_n = i$, $\text{toestand}_{n-1} = h$ wordt samengenomen tot $\text{Toestand}_n = (h, i)$. De toekomstige Toestand moet ook op deze manier beschreven kunnen worden —en dat lukt, want de huidige toestand, dus de eerste uit het benodigde paar, is bekend. We gebruikten deze vondst al eerder en wel direkt bij de introduktie van Markovketens in het vorige hoofdstuk in het voorbeeld 'geheugen van twee stappen' (§9.3.1).

De overgangskans naar toestand_j wordt nu

$$Pr(\text{Toestand}_{n+1} = (i, j) \mid \text{Toestand}_n = (h, i))$$

Daarmee is de kans uit de Markovketen van de tweede orde geschreven als een overgangskans in een eerste orde keten met een uitgebreidere verzameling toestanden: alle paren van toestanden uit de tweede orde keten.

Ook k-de orde Markovketens gaan over in eerste orde Markovketens door het invoeren van *sequenties van toestanden* uit de k-de orde keten als nieuwe toestanden. De sequenties moeten in tijdvolgorde staan. De beperking tot eerste orde Markovketens is dus geen essentiële beperking!

10.4.2. voorbeeld: hogere orde keten ingeperkt tot eerste orde

We willen eens in een concreet geval nagaan hoe de betrekkingen tussen sequenties van toestanden en de overgangsmatrices van hogere en eerste orde kunnen liggen. We gaan daarbij uit van meetgegevens over het gebruik van het UNIX-filesysteem. De specificaties (meetgegevens) zijn (bijvoorbeeld) als volgt.

In een UNIXsysteem zijn 3 filesystemen in gebruik. Naast het root filesysteem /root zijn ook de filesystemen /tmp en /usr beschikbaar ('gemount'). Het UNIX-systeem draait op een microcomputer, alle drie filesystemen staan op dezelfde Winchesterdisk. Voor de te verwerken werklast is nagegaan in welke volgorde deze filesystemen (gemiddeld) worden benaderd. Op een acces op /tmp volgt gemiddeld in 4 van de 5 gevallen opnieuw een acces op /tmp; er kan ook worden overgegaan naar /root of /usr. Beide mogelijkheden blijken gemiddeld even vaak voor te komen. Een acces op /tmp wordt dus gemiddeld in 1 op de 10 gevallen gevolgd door een acces op /usr. Op een acces op /root volgt in gemiddeld 1 op de 5 gevallen opnieuw een acces op /root, anders volgt een acces op /usr. Als een

acces op /usr wordt voorafgegaan door een acces op /tmp, volgt in gemiddeld 4 op de 5 gevallen opnieuw een acces op /usr, in 1 op de 10 gevallen een acces op /tmp en in 1 op de 10 gevallen een acces op /root. Als een acces op /usr wordt voorafgegaan door een acces op /root zijn deze relatieve frequenties respectievelijk 3/5, 1/5 en 1/5. Als er voor /usr twee accessen achter elkaar zijn geweest, liggen de relatieve frequenties nog anders, en wel als 2/5, 2/5 en 1/5.

Uit deze gegevens blijkt dat het voor de accessen vanuit /usr uitmaakt wat het daaraan voorafgaande acces was. Als dit naar /tmp was, zal /tmp in 1/10 van de gevallen worden benaderd, als dit /root was in 1/5 en als de vorige overgang ook naar /usr was in 2/5 van de gevallen. Bij de accessen vanuit /tmp en /root zijn zulke verschillen niet waargenomen. Na /tmp volgt /tmp in gemiddeld 4/5 van de gevallen, na /root volgt (vrijwel) nooit /tmp.

De relevante patronen, die in de opvolging van de filesystemen moeten worden onderscheiden, zijn daarom kennelijk (acces op) /tmp, /root, en (acces op, gevolgd door acces op) /tmp,/usr respectievelijk /root,/usr respectievelijk /usr,/usr.

Dit zijn dan ook de toestanden die we in de Markovketen gaan onderscheiden. De toestanden zijn: laatste acces was naar het /tmp filesysteem, naar het /root filesysteem en voorlaatste acces was naar het /tmp filesysteem waarna een acces op /usr volgde, voorlaatste acces was naar het /root filesysteem waarna een acces op /usr volgde en voorlaatste acces was naar het /usr filesysteem waarna een acces op /usr volgde. Bij /usr hebben we een sequentie van twee accessen nodig, bij /tmp en /root volstaan we met één acces. De overgangen tussen de filesystemen markeren hier de toestanden. We noteren de toestanden als /tmp, /root, tmp/usr, root/usr en usr/usr. De overgangsmatrix tussen deze toestanden staat in figuur 10.3 (vergelijk §9.7).

	/tmp	*/root*	*tmp/usr*	*root/usr*	*usr/usr*
/tmp	4/5	1/10	1/10	0	0
/root	0	1/5	0	4/5	0
tmp/usr	1/10	1/10	0	0	4/5
root/usr	1/5	1/5	0	0	3/5
usr/usr	2/5	1/5	0	0	2/5

Figuur 10.3. Kansen op overgangen tussen filesystemen.

De kansen π op de toestanden in de stationaire fase volgen door $\pi = \pi \mathbf{P}$ op te lossen, met voor **P** de overgangsmatrix uit figuur 10.3. De uitkomsten zijn, in volgorde van optreden in de matrix, ($\pi_{/tmp}$ = 720/1421, $\pi_{/root}$ = 205/1421, $\pi_{tmp/usr}$ = 72/1421, $\pi_{root/usr}$ = 164/1421, $\pi_{usr/usr}$ = 260/1421) of (0.507, 0.144, 0.051, 0.115, 0.183).

Accessen op de filesystemen /tmp, /root en /usr zullen dus niet gemiddeld even vaak voorkomen. Zo zal /usr (72+164+260)/205 = 2.42 keer zo vaak worden

gebruikt worden als /root. Het deel (164/1421 = 0.115) van de accessen op de drie filesystemen is een acces op /usr, voorafgegaan door een acces op /root. Dat is direkt af te lezen aan de kansen op de lange termijn. Ook het gespiegelde deel, waarin /usr gevolgd wordt door /root, is te bepalen. Dit is $\pi_{tmp/usr} \times 1/10 + \pi_{root/usr} \times 1/5 + \pi_{usr/usr} \times 1/5 = 0.065$. Een acces op /usr zal gevolgd worden door òf een acces op /tmp, òf een acces op /root, òf een acces op /usr. Het deel waarin een acces op /root volgt is gemiddeld

$$\frac{(\pi_{tmp/usr} \times 1/10 + \pi_{root/usr} \times 1/5 + \pi_{usr/usr} \times 1/5)}{(\pi_{tmp/usr} + \pi_{root/usr} + \pi_{usr/usr})} = 23/124 = 0.185$$

En dan nu het interessantste aspekt.

De processen uit de werklast pendelen heen en weer tussen de toestanden: 'laatste acces was op /tmp', 'laatste acces was op /root' en 'laatste acces was op /usr'. Kortweg, ze wisselen tussen de toestanden /tmp, /root en /usr. Bestaat er een overgangsmatrix tussen deze toestanden en hoe ziet die er uit?

Voor de toestanden /tmp en /root kunnen de overgangskansen uit de gegevens worden afgeleid, ze zijn (4/5, 1/10, 1/10) en (0, 1/5, 4/5). De gegevens maken ook duidelijk dat de toestand /usr overgangskansen heeft, die afhangen van de direkt daaraan voorafgaande toestand; de toestand heeft een geheugen van twee stappen. Daarom gebruikten we de drie toestanden tmp/usr, root/usr en usr/usr. Maar als we nu toch enkel /usr verder als toestand willen invoeren? Over de drie mogelijkheden voor de voorgaande toestand kan toch gemiddeld worden?

Inderdaad, dat kan. Maar de primitieve weging 1/3, 1/3, 1/3 is onjuist. Er moeten gewichten worden genomen, die evenredig zijn met de relatieve frequentie van het optreden als voorganger! Alleen op die manier ontstaat een heuse eerste orde Markovketen. Dit laten een operationele beschouwing of de rekenregels voor voorwaardelijke kansen, snel zien.

De relatieve frequenties van optreden als voorganger volgen niet rechtstreeks uit de gegevens. Ze zijn bekend zodra de keten met de 5 toestanden is doorgerekend. De relatieve frequenties zijn volgens de stationaire kansen als 72:164:260 of 18:41:65. De overgangskans om van /usr naar /root te gaan is

$$\frac{\pi_{tmp/usr} \times 1/10 + \pi_{root/usr} \times 1/5 + \pi_{usr/usr} \times 1/5}{\pi_{tmp/usr} + \pi_{root/usr} + \pi_{usr/usr}}$$

Zojuist vonden we voor deze kans 23/124. Op dezelfde manier rekenen we de overgangskansen van /usr naar /tmp en naar /usr uit. We vinden dan de overgangsmatrix uit figuur 10.4 voor de eerste orde Markovketen met als toestanden /tmp, /root en /usr (de effektieve 1-staps overgangsmatrix). De stationaire kansen $\pi = (0.507, 0.144, 0.349)$ op het optreden van accessen op /tmp, /root en /usr volgen hieruit met $\pi = \pi \mathbf{P}$, met deze 3-bij-3 matrix **P**. Het zijn natuurlijk niets anders dan de π_{tmp}, π_{root} en $(\pi_{tmp/usr} + \pi_{root/usr} + \pi_{usr/usr})$ van (720/1421, 205/1421, 496/1421),

	/tmp	*/root*	*/usr*
/tmp	4/5	1/10	1/10
/root	0	1/5	4/5
/usr	9/31	23/124	65/124

Figuur 10.4. Effektieve 1-staps overgangsmatrix.

die we al berekend hebben. Dit blijkt expliciet door in de matrix **P** de uitdrukkingen voor de overgangskansen niet numeriek als 9/31, 23/124 en 65/124 in te vullen, maar gewoon in de overgangskansen de uitdrukkingen met $\pi_{tmp/usr}$, $\pi_{root/usr}$, en $\pi_{usr/usr}$ te laten staan en dan het stelsel $\pi = \pi \mathbf{P}$ op te lossen.

De overgangsmatrix met de toestanden /tmp, /root en /usr bevat minder detailinformatie dan de overgangsmatrix voor de 5 toestanden. De relatieve frequentie van de sequentie usr/root is 0.349×23/124 en dat is weer 0.065, en van de sequentie root/usr 0.144×4/5 en dat is weer 0.115. Maar het relatieve voorkomen van de sequentie /tmp, gevolgd door /usr, gevolgd door /tmp, zou zijn (0.507 = 720/1421)×1/10×9/31 en niet de waarde van 72/1421×1/10 volgens de overgangsmatrix met 5 toestanden. De middeling bij de overgangskansen vanuit /usr gooit hier roet in het eten.

Merk op dat we de overgangsmatrix voor deze tot een eerste-orde keten 'ingeperkte' hogere orde keten pas kunnen uitrekenen nadat we de hogere orde keten hebben opgelost. Want de gegevens staan niet toe de eerste-orde matrix zonder meer op te stellen, ze geven alleen aan hoe de hogere orde overgangsmatrix eruit ziet.

10.5. VOORBEELD: LOCAL RANDOM REPLACEMENT

Het komt nogal eens voor dat bij het aangeven van een toestand niet met het opgeven van de waarde van een voor de hand liggende parameter kan worden volstaan. Het is dan zaak uit te zoeken hoe de relevante toestanden precies moeten worden gedefinieerd. Dit speelt in het volgende voorbeeld.

Een operatingsysteem werkt met paginering. Ieder programma krijgt een aantal geheugenpagina's ter beschikking om zijn pagina's te herbergen als ze in het geheugen ('in core') zijn (deze page frames vormen zijn partitie). De partitiegrootte is meestal veel kleiner dan de grootte van het programma aan code en data. Als bij het executeren van het programma een referentie optreedt naar een andere dan de pagina waarop gewerkt wordt, vindt een paginaoverstap naar een andere pagina plaats (page change). Heel vaak betekent een paginaoverstap niets anders dan dat op de andere pagina wordt verder gegaan, maar soms is deze pagina niet in het geheugen aanwezig, dan ontbreekt deze in de partitie van het betrokken programma. Er treedt in zo'n geval een page fault op (de overstap genereert een page fault) en het executeren van het programma wordt gestaakt. Een I/O-apparaat krijgt opdracht de ontbrekende pagina in de partitie te laden. Dit gaat

ten koste van een reeds in de partitie aanwezige pagina. Na enige tijd komt het programma weer aan de beurt bij de CPU en het executeren gaat verder op de verse pagina.

De prestaties van het paging-mechanisme worden duidelijk beïnvloed door het vervangingsalgoritme, dat uitmaakt welke nu aanwezige pagina plaats moet maken voor de nieuwkomer. Het klassieke algoritme voor de vervanging is LRU (LEAST RECENTLY USED). Voor het werken met dit algoritme is het nodig voor elke pagina in het geheugen bij te houden hoeveel paginaoverstappen geleden voor het laatst aan deze pagina gerefereerd is (er een paginaoverstap naar deze pagina plaats vond). Daarmee is steeds bekend voor welke pagina deze referentie het verst in het verleden ligt, dit is de *least recently used* pagina. Het is deze pagina die onder LRU plaats moet maken voor de nieuwkomer.

Een eenvoudiger strategie is RANDOM REPLACEMENT. Er wordt dan geen nadere informatie bijgehouden. Bij local random replacement heeft elk van de in de partitie aanwezige pagina's dezelfde kans dat hij het veld moet ruimen. We willen de eigenschappen van deze strategie nader onderzoeken. Doel is na te gaan welke pagina's meestal in het geheugen zitten en welke niet.

Dezelfde vervangingsalgoritmen worden ook toegepast als de jongste resultaten van de accessen op een device in een '*cache*' bewaard blijven, zodat steeds het 'relevante' deel van de inhoud van de resource in de cache aanwezig is. Wat we straks leren kan dus uitstijgen boven de typische context van gepagineerd gebruik van geheugen.

Het gedrag onder het algoritme zal sterk afhangen van het overstappatroon, dus van het referentiepatroon in de code van het programma. Er wordt sinds lang gezocht naar algemene karakteristieken van dit patroon; meestal is er een duidelijke tendens naar 'lokale' referenties, maar deze ontbreekt soms in moderne gestructureerde (met generatoren aangemaakte) programmatuur, die veel systeemfunkties gebruikt.

We nemen aan dat een bepaald programma 4 pagina's groot is en een ruimte ter beschikking heeft van 2 pagina's (we nemen een beetje onnozele waarden). De partitiegrootte is dus 2 en er is steeds maar de helft van het programma in het geheugen. De pagina's die in het geheugen zijn, bepalen de toestand van het systeem. De toestand waarin het verwerken door de CPU plaats vindt binnen pagina 2 (of aanstonds zal plaatsvinden, zodra het programma in executie genomen wordt; preciezer: de program counter (PC) wijst naar een adres in pagina 2) noteren we als 2(4) wanneer naast pagina 2 ook nog pagina 4 in de partitie aanwezig is; de toestand is 2(3) als niet 4 maar 3 aanwezig is, enzovoort.

Uit metingen aan het referentiepatroon wordt informatie verkregen over het overstappatroon. Als pagina 1 door de CPU verwerkt wordt, is de volgende paginaoverstap in 3 van de 5 gevallen naar pagina 4, in 1 van de 5 gevallen naar pagina 2 en in 1 van de 5 gevallen naar pagina 3. Analoog voor de andere pagina's. Van de vier pagina's is kennelijk vooral de vierde belangrijk. Het overstappatroon kan worden weergegeven door de matrix uit figuur 10.5. Dit is echter niet de relevante overgangsmatrix!

$$\begin{bmatrix} 0 & 1/5 & 1/5 & 3/5 \\ 1/5 & 0 & 1/5 & 3/5 \\ 1/5 & 1/5 & 0 & 3/5 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Figuur 10.5. Kansen op overgangen naar andere pagina. Element *ij* is de kans dat er overgestapt wordt van pagina *i* naar pagina *j*.

Er zijn in dit systeem 12 verschillende toestanden $i(j)$. De waarden van de elementen van de overgangsmatrix tussen deze toestanden volgen uit het overstappatroon en de vervangingsstrategie. Het element (2(3), 4(2)) bijvoorbeeld wordt als volgt bepaald. Tijdens het executeren van pagina 2 wordt de afwezige pagina 4 gerefereerd, er is dus een page fault. De kans dat de verwerking door de CPU op deze manier wordt vervolgd is volgens de overstaptabel 3/5. Om voor 4 plaats te maken moet of 3 of 2 wijken. De kans dat 3 het veld ruimt is bij random replacement 1 op 2. Daardoor is de totale kans dat 4(2) de volgende toestand is, gelijk aan $3/5 \times 1/2 = 3/10$; in de overgangsmatrix staat dus op de positie (2(3), 4(2)) de waarde 3/10. Net zo is het element (2(3), 4(3)) gelijk aan 3/10.

De stationaire toestandsvergelijking die bij de overdrachtsmatrix hoort geeft aan welke toestanden gemiddeld voorkomen. Het blijkt dat $\pi_{i(j)} = 5/96$, $\pi_{i(4)} = 5/48$ en $\pi_{4(i)} = 1/8$, waarin i en j 1, 2 of 3 zijn. Dus in 11 op de 16 optredende partitievullingen is pagina 4 in het geheugen aanwezig en in 3 op de 8 heeft pagina 4 de executiebeurt en wordt er bij het executeren op deze pagina gewerkt.

We kunnen de vraagstelling nog verder uitbouwen door ook tijdsduren in te brengen, zoals de lifetime: de gemiddelde tussenpoos tussen twee page faults. Deze zal in het algemeen per pagina anders liggen. Kennis over de verhouding van deze lifetimes maakt het mogelijk over te stappen van de kansen op optreden π_i op de kansen op het aantreffen van de toestanden p_i: gebruik de basisrelatie uit het hoofdstuk BEURTEN EN KANSEN.

Uit de stationaire kansen kan verder de frequentie waarmee page faults optreden worden berekend. Ook is na te gaan hoe sterk deze frequentie (de page fault rate) daalt door de veel gerefereerde pagina 4 permanent in het geheugen te laten.

10.6. SAMENVATTING

In dit hoofdstuk bekeken we het algemene nut van Markovketens. Markovketens blijken eenvoudige, plezierige en krachtige hulpmiddelen te zijn om een stapsgewijze voortgang te beschrijven. Ze maken het mogelijk aan te geven in welke toestanden systemen uiteindelijk terecht komen of tussen welke toestanden systemen uiteindelijk heen en weer pendelen, en hoe vaak die toestanden relatief worden aangedaan. De stationaire toestand bij ergodische Markovketens ontstaat door 'uitmiddelen'. Systemen met geheugen (hogere orde Markovketens) kunnen worden herschreven tot systemen 'zonder geheugen' (eerste orde Markovketens)

door de voorgeschiedenis in de definitie van de toestand op te nemen (§10.4.1) of door te middelen over de voorgeschiedenis (§10.4.2).

10.7. OPGAVEN

opgave 10.1: periodiek

Een systeem komt in twee toestanden, A en B. De overgangskansen worden bepaald door de huidige toestand en de voorafgaande toestand. De situatie met als opeenvolgende toestanden A en B geven we aan met AB. Er is bekend dat AB optreedt vanuit AA met een kans a, BB vanuit AB met een kans b, AA vanuit BA met een kans c en BA vanuit BB met een kans d (a, b, c en d zijn ongelijk aan nul).

- Bepaal het kwadraat van de overgangsmatrix. Komt elke toestand ooit voor?
- Geef aan hoe groot de kans is dat situatie AB is bereikt na 4, respectievelijk 8 overgangen, uitgaande van situatie AA.

Bekijk vervolgens het speciale geval dat a, b, c en d 1 zijn (na 4 stappen is ieder terug).

opgave 10.2: absorberend

Beschrijf de routering van het verkeer in een aantal door gateways gekoppelde lokale netwerken met behulp van een algemene Markovketen.

opgave 10.3: communicatie

De president van een zeker land vertelt een intieme vriend of hij al dan niet betrokken was bij besluit 'X'. Deze vriend vertelt het nieuws door aan een ander en deze weèr aan de volgende persoon, enzovoort. We nemen aan dat er een kans a bestaat dat zo'n persoon een als 'ja' ontvangen bericht als 'nee' doorvertelt en een kans b dat een als 'nee' ontvangen bericht als 'ja' wordt doorgegeven. De uitgangstoestand is de informatie, die door de president is gegeven.

- Bepaal de overgangsmatrix $\mathbf{P}^n$. Wat gebeurt er als n nadert tot oneindig? Interpreteer het resultaat.
- Neem $a = 1/10$, $b = 1/20$. Bepaal voor dit geval de overgangsmatrix naar de stationaire fase met de formule uit §10.3. Geef een uitdrukking voor het verschil tussen de kans op 'ja' na n stappen en de kans op 'ja' in de stationaire toestand (§10.3.1).

(wat er uiteindelijk als informatie wordt doorgegeven hangt niet af van de keus van de president, maar van de kwaliteit van het transmissiemedium: de kansen op ja/nee zijn als b/a; a en b beschrijven hoe slecht de boodschap wordt doorgegeven.

Dit uitvlakken van informatie doet zich bij communicatie altijd voor, ook bij automatische transmissie van informatie over een computernetwerk: zorg voor weinig schakels).

opgave 10.4: MTBF

Bij computersystemen wordt gewoonlijk bijgehouden hoeveel tijd er verstrijkt tussen twee opvolgende storingen. Deze tussenpozen zijn gemiddeld vrij lang in een periode dat het systeem 'goed' draait, maar af en toe komen er 'slechte' perioden voor waarin storingen vlak na elkaar komen. Storingen zijn 'bursty': nu eens zijn er telkens problemen, dan weer is er heel lang alleen incidenteel een moeilijkheid. De verwachtingswaarde van de tussenpozen (TBF) wordt de MEAN TIME BETWEEN FAILURES (MTBF) genoemd.

Wanneer het systeem betrekkelijk snel na de laatste reparatie plat is gegaan, het draait 'slecht', is er een kans van 1 op 5 dat het systeem na de komende reparatie nu langere tijd zal funktioneren, dus 'goed' gaat draaien. Wanneer het systeem goed draait en er treedt een storing op, is er een kans van 1 op 10 dat het daarna 'slecht' gaat draaien.

- Onderscheid de toestanden 'slecht' en 'goed' en stel de overgangsmatrix tussen deze toestanden op. Bepaal de stationaire toestandsvector.
- Bepaal voor de stationaire fase hoeveel maal de toestand 'slecht' gemiddeld optreedt per toestand 'goed', dus ook hoeveel 'slechte' TBF's er gemiddeld zitten tussen twee opvolgende 'goede' TBF's (1/2).
- Bepaal ook het aantal 'slechte' TBF's dat gemiddeld achter elkaar optreedt. Wat is het verschil tussen dit antwoord en het antwoord op de voorgaande vraag?
- De 'slechte' TBF's duren gemiddeld 150 uur, de 'goede' TBF's duren gemiddeld viermaal zo lang. Een medewerker start een project op de dag dat het systeem weer net gerepareerd is. Hij weet niet of het systeem goed of slecht draait. Wat is zijn verwachting van de tijd tot de volgende storing (450), wat neemt hij aan?
- Tijdens de koffie krijgt de medewerker echter te horen dat het systeem voor de laatste reparatie een lange TBF achter de rug heeft. Wat wordt nu zijn verwachting van de tijd tot de volgende storing (555)? (hij neemt aan dat een lange TBF betekent dat het systeem 'goed' draait.)
- Uit het verleden weet men dat zowel de korte als de lange TBF's een variantie van 10.000 hebben. Wat zijn de variatiecoëfficiënten van de 'slechte' en de 'goede' TBF's ? Een controleur komt op willekeurige momenten binnenlopen en kijkt dan hoelang het systeem al draait na de meest recente reparatie. Wat vindt hij gemiddeld (286.1)?

opgave 10.5: verdeling seeks

Op een schijf staan drie systeemfiles, die we de datasets A, B en C noemen. De ruimte op de schijf is opgedeeld in cylinders. Dataset A beslaat de cylinders 100-124, dataset B de cylinders 200-224 en dataset C de cylinders 250-274. We bekijken het verkeer op deze schijf en beperken ons daarbij tot de beweging van de lees-schrijfkop over de schijf, dus tot de 'seek'-aktie. Telkens als een funktie uit het operatingsysteem bepaalde gegevens van dataset A nodig heeft, vindt een acces op de schijf plaats en gaat de kop naar de betrokken cylinder onder de cylinders 100-124 van dataset A; idem bij B en C. Het patroon van accessen op de datasets blijkt als volgt te kunnen worden omschreven:

> Op een acces op een dataset volgt in de helft van de gevallen als volgend acces een acces op dezelfde dataset.
>
> Op een acces op dataset A volgt in een kwart van de gevallen een acces op dataset C; op een acces op dataset B volgt in een zesde deel van de gevallen een acces op dataset A; op een acces op dataset C volgt in een twaalfde deel van de gevallen een acces op dataset A.

Als een acces op een dataset gevolgd wordt door een acces op dezelfde dataset moet de kop een seek doen, die gemiddeld 5 msek. duurt. Wanneer een acces op een dataset gevolgd wordt door een acces op een andere dataset duurt de seek langer en wel gemiddeld 4 msek. (voor het 'op gang komen') + 0.1 msek. maal de gemiddelde afstand in aantallen te overbruggen cylinders. De seek van dataset A naar dataset B duurt bijvoorbeeld gemiddeld $4 + 100 \times 0.1 = 14$ msek.

- Zojuist kwam een acces binnen voor dataset A. Bepaal hoe groot de kans is dat het 10-de acces na dit acces ook weer voor dataset A zal zijn. Hoe groot is de kans dat een sequentie 'acces op B, gevolgd door acces op B' direkt komt na de sequentie 'acces op A, gevolgd door acces op B'? Hoe groot is de kans dat, als het derde acces voor dataset A blijkt te zijn, ook het vijfde acces voor dataset A zal zijn?
- Welk deel van de accessen op de schijf is voor dataset A en welk deel voor dataset B (16/79, 33/79)?
- Hoe groot is de gemiddelde seektijd, die optreedt als een acces op B gevolgd wordt door een acces op C? Hoe groot is de gemiddelde seektijd bij de accessen, die rechtstreeks (als eerstvolgende) volgen op een acces op B (47/6)?
- Hoe groot is de gemiddelde seektijd bij de accessen op de schijf (8.42)?
- Op een lukraak moment komt er een acces binnen. De lees-schrijfkop blijkt bezig te zijn. Hoelang duurt het gemiddeld nog voordat deze 'seek' aktie is afgelopen? Wat wordt dan aangenomen?

opgave 10.6: Send-and-Wait

De processen 'Links' en 'Rechts' uit een computernetwerk houden voortdurend contact; ze sturen elkaar aanhoudend informaties (info's) en aanvragen voor informatie (requests). De berichten die tussen hen over een full-duplex verbinding worden uitgewisseld zijn uitsluitend van het type info of van het type request.

Het berichtenverkeer wordt volgens het Send-and-Wait protocol afgewikkeld. Er wordt pas met het zenden van het volgende bericht begonnen als de aankomst van het vorige bericht door de partner bevestigd is met een 'ack' (of met een 'nack' als er iets mis ging). Links en Rechts doorlopen zodoende steeds de volgende cyclus. Ze beginnen tegelijk met het verzenden van hun volgende bericht (een info of een request). Een van beiden zendt het kortste bericht en gaat wachten tot de ander klaar is met het verzenden. Zodra beiden gereed zijn sturen ze elkaar een ack (eventueel een nack) om de goede overkomst van het bericht te melden.

De toestanden bij de communicatie zijn:

RR: in deze cyclus zenden Links en Rechts een request,
RI: idem Links een request, Rechts een info,
IR: idem Links een info, Rechts een request,
II: idem Links een info, Rechts een info.

Als Links een request van Rechts binnenkrijgt, zendt het Rechts in de volgende cyclus in gemiddeld 1 op de 3 gevallen een request. Als Links een info van Rechts binnenkrijgt, zendt het in de volgende cyclus een request naar Rechts. Als Rechts een request van Links binnenkrijgt, zendt het in de volgende cyclus in gemiddeld 1 op de 4 gevallen een request naar Links. Als Rechts een info van Links binnenkrijgt, zendt het in de volgende cyclus een request naar Links.

- Bepaal de overgangsmatrix met de kansen op de overgangen tussen de relevante toestanden. Controleer dat de stationaire toestandsvector $(1/3, 1/3, 1/6, 1/6)$ is. In gemiddeld welk deel van de cycli sturen Links en Rechts elkaar informatie (info's)?
- De ack's (en de nack's) duren altijd precies 1 msek. Bij proces Links duurt een request altijd 15 msek. en een info altijd 35 msek. Bij proces Rechts duurt een request altijd 15 msek. en een info altijd 20 msek. Bepaal de gemiddelde duur van een cyclus. Bepaal hoe groot de kans is, dat op een willekeurig moment zowel Links als Rechts bezig zijn met het versturen van een ack (nack) op een net ontvangen info $(1/146)$.
- Proces Rechts wordt op een lukraak moment geïnterrumpeerd. Hoelang zal het vanaf zo'n moment nog gemiddeld duren tot Rechts zijn volgende bericht gaat versturen?
- Bij het Send-and-Wait protocol gaat er lijncapaciteit verloren doordat de ack's tijd kosten en doordat er op de ack's werkeloos wordt gewacht. Probeer in dit geval aan te geven of deze verliezen relatief groot zijn.

11

Markovketens en evenwichtsvergelijkingen

11.1. INLEIDING

In dit hoofdstuk bespreken we —eindelijk— de operationele afleiding van de evenwichtsvergelijkingen

$$\pi = \pi \mathbf{P}$$

Dit geeft een nieuwe kijk op deze vergelijkingen, het blijkt dat ze ook gevonden kunnen worden zonder eerst de overgangsmatrix op te stellen. De operationele afleiding houdt verband met het principe van *'binnendruppelen is weglekken'*. Dit principe geeft aan dat in een systeem evenwicht heerst. Het kan op alle toestanden van het systeem worden toegepast en legt de evenwichtsvergelijkingen zonder meer vast. Dit evenwichtsprincipe kan bovendien bij systemen met een continu tijdsverloop worden gebruikt.

Dit hoofdstuk is wat beschouwelijk van aard en is bedoeld voor wie geïnteresseerd is in de operationele aanpak en de betekenis van evenwichtsvergelijkingen. Het is op zich een intermezzo, kennis er van is niet nodig om de andere hoofdstukken te begrijpen. Doel is de blik te verruimen om de resultaten van Markovketens te kunnen generaliseren naar systemen met een continu verloop in de tijd, zoals *Markovprocessen* en wachttijdensystemen.

We gaan eerst in op de vraag welke data nodig zijn om een berekening met Markovketens op te zetten. Daarna leiden we de operationele vorm van de evenwichtsvergelijkingen af en staan kort stil bij het belang van zo'n 'afleiding'. We bepalen de evenwichtsvergelijkingen voor onze eerdere voorbeelden met behulp van hun toestandsdiagrammen.

11.2. OPERATIONELE ANALYSE BIJ MARKOVKETENS

11.2.1. de relatie tussen overgangskansen en kansen op optreden

Bij het onderwerp Markovketens is de operationele aanpak nog niet benut. Bij Markovketens komen zowel de kracht als de zwakte van de operationele aanpak naar voren, daarom was het zaak eerst de uitspraken van de Markovtheorie te leren kennen.

De operationele aanpak wordt geïllustreerd aan de hand van het voorbeeld uit MARKOVKETENS EN COMPUTERCONFIGURATIES (§9.9). Om de hoeveelheid indices klein te houden vereenvoudigen we het voorbeeld tot een configuratie met als devices een aantal terminals, een CPU en een (samengesteld) I/O-randapparaat.

Zowel de overgangskansen als de kansen op optreden blijken in principe te worden bepaald uit de gemeten aantallen overgangen tussen de devices. Dus èn de vertakkingsverhoudingen èn de bezoekfrequenties worden uit dezelfde data berekend.

In de meetduur *MEETDUUR* zullen de aantallen overgangen van het ene device naar het andere worden bijgehouden. De teller voor het aantal overgangen van CPU naar I/O wordt verhoogd als een usercommando overgaat van de toestand 'bij de CPU' naar de toestand 'bij het I/O-apparaat', doordat er een I/O-opdracht is gegenereerd. Na afloop van het meten zijn bekend:

- Het aantal overgangen in de meetduur van de CPU naar het I/O-apparaat, respectievelijk de gebruikers (vervanging usercommando's): $DOOR_{CPU\,I/O}$ en $DOOR_{CPU\,gebruiker}$.
- Idem van I/O naar CPU en gebruikers: $DOOR_{I/O\,CPU}$ en $DOOR_{I/O\,gebruiker}$.
- Idem van gebruikers naar CPU en I/O (nieuwe usercommando's): $DOOR_{gebruiker\,CPU}$ en $DOOR_{gebruiker\,I/O}$.

Het aantal $DOOR_{gebruiker\,CPU}$ commando's begint met een CPU-acces en het aantal $DOOR_{gebruiker\,I/O}$ begint met een I/O-behandeling (we nemen de aktiviteiten van het operatingsysteem zoals 'spooling' maar niet separaat mee).
Het aantal overgangen vanaf de CPU is

$$UIT_{CPU} = DOOR_{CPU\,I/O} + DOOR_{CPU\,gebruiker}$$

Het aantal overgangen naar de CPU is

$$IN_{CPU} = DOOR_{I/O\,CPU} + DOOR_{gebruiker\,CPU}$$

Deze aantallen zullen vrijwel gelijk zijn:

$$UIT_{CPU} \approx IN_{CPU}$$

Het verschil tussen IN_{CPU} en UIT_{CPU} is een *randeffekt*, het kan hoogstens gelijk zijn aan het grootste aantal usercommando's dat tijdens de meetduur tegelijkertijd bij de CPU is.

Voor de andere devices geldt analoog

$$IN_{gebruiker} = DOOR_{CPU\,gebruiker} + DOOR_{I/O\,gebruiker}$$

$$\approx UIT_{gebruiker} = DOOR_{gebruiker\,CPU} + DOOR_{gebruiker\,I/O}$$

$$IN_{I/O} = DOOR_{CPU\,I/O} + DOOR_{gebruiker\,I/O}$$

$$\approx UIT_{I/O} = DOOR_{I/O\,CPU} + DOOR_{I/O\,gebruiker}$$

Deze 'gelijkheden' tussen de *IN*'s en de *UIT*'s zijn operationele 'behoudsrelaties', zoals we die in de voorgaande hoofdstukken al vaak zijn tegengekomen, bijvoorbeeld in §5.3. De behoudsrelaties geven in dit geval aan dat er geen commando's verdwijnen of opgepot worden. Bij de eerdere afleidingen van basisrelaties uit operationele schatters stapten we nogal vlot over randeffekten heen (§4.4, §5.4, §7.4).

Aan de UIT_{CPU} overgangen vanaf de CPU is een bezoek aan de CPU voorafgegaan, aan de $UIT_{I/O}$ overgangen vanaf de I/O een bezoek aan het I/O-apparaat en aan de $UIT_{gebruiker}$ overgangen vanaf de terminals een 'bezoek' aan de terminals. In totaal wordt er *UIT* keer genoteerd dat een usercommando een toestand aandoet, met

$$UIT = UIT_{CPU} + UIT_{I/O} + UIT_{gebruiker}$$

De relatieve frequentie waarmee de CPU wordt aangedaan, is dus UIT_{CPU}/UIT. De operationele schatter voor de kans π_{CPU} dat de CPU de beurt heeft, is dan ook

$$'\pi_{CPU} = UIT_{CPU}/UIT$$

en analoog voor $'\pi_{I/O}$ en $'\pi_{gebruiker}$.

Uit het waarnemingsmateriaal bestaande uit de *DOOR*'s volgen ook, heel direkt, de gemeten waarden voor de vertakkingsverhoudingen. Zo gaat bij de CPU het deel $DOOR_{CPU\,I/O} / UIT_{CPU}$ van de overgangen naar het I/O-apparaat. De operationele schatter voor $\mathbf{P}_{CPU\,I/O}$ zal zijn:

$$'\mathbf{P}_{CPU\,I/O} = DOOR_{CPU\,I/O} / UIT_{CPU}$$

en idem voor de andere vertakkingen, bijvoorbeeld

$$'\mathbf{P}_{I/O\,CPU} = DOOR_{I/O\,CPU} / UIT_{I/O}$$

De som van de waarden van de operationele schatters bij de vertakkingsverhouding voor de CPU is netjes precies 1, omdat UIT_{CPU} de som is van de *DOOR*'s bij vertrek vanaf de CPU:

$$'\mathbf{P}_{CPU\,I/O} + {}'\mathbf{P}_{CPU\,gebruiker} =$$

$$= DOOR_{CPU\,I/O} / UIT_{CPU} + DOOR_{CPU\,gebruiker} / UIT_{CPU}$$

$$= UIT_{CPU} / UIT_{CPU} = 1$$

Net zo bij het I/O-apparaat en de gebruikers, dus in totaal

$$'\mathbf{P}_{CPU\,I/O} + {}'\mathbf{P}_{CPU\,gebruiker} = 1$$

$$'\mathbf{P}_{I/O\,CPU} + {}'\mathbf{P}_{I/O\,gebruiker} = 1$$

$$'\mathbf{P}_{gebruiker\,CPU} + {}'\mathbf{P}_{gebruiker\,I/O} = 1$$

De diagonaalelementen in $'\mathbf{P}$ zijn nul, omdat in de gang van de usercommando's geen rechtstreekse overgangen voorkomen van een device naar hetzelfde device. Er staat dat $'\mathbf{P}\,\mathbf{I}(3) = \mathbf{I}(3)$. Dit is analoog aan (9.1) uit §9.3.

Er ontstaan behoorlijk schokkende betrekkingen als de operationele schatters worden ingebracht in de operationele behoudsrelaties.

Herschrijven met de operationele schatters voor de vertakkingsverhoudingen (de overgangskansen) levert bij de CPU bijvoorbeeld dat IN_{CPU} ook is:

$$IN_{CPU} = DOOR_{I/O\,CPU} + DOOR_{gebruiker\,CPU}$$

$$IN_{CPU} = {}'\mathbf{P}_{I/O\,CPU}\, UIT_{I/O} + {}'\mathbf{P}_{gebruiker\,CPU}\, UIT_{gebruiker}$$

Als de randeffekten kunnen worden verwaarloosd, zou $IN_{CPU} = UIT_{CPU}$ zijn, zodat

$$UIT_{CPU} = {}'\mathbf{P}_{I/O\,CPU}\, UIT_{I/O} + {}'\mathbf{P}_{gebruiker\,CPU}\, UIT_{gebruiker}$$

Na delen door *UIT* staat er als relatie tussen de operationele schatters:

$${}'\pi_{CPU} = {}'\pi_{I/O}\, {}'\mathbf{P}_{I/O\,CPU} + {}'\pi_{gebruiker}\, {}'\mathbf{P}_{gebruiker\,CPU}$$

Net zo gaat het bij de I/O en de gebruikers:

$${}'\pi_{I/O} = {}'\pi_{CPU}\, {}'\mathbf{P}_{CPU\,I/O} + {}'\pi_{gebruiker}\, {}'\mathbf{P}_{gebruiker\,I/O}$$

$${}'\pi_{gebruiker} = {}'\pi_{CPU}\, {}'\mathbf{P}_{CPU\,gebruiker} + {}'\pi_{I/O}\, {}'\mathbf{P}_{I/O\,gebruiker}$$

De relaties tussen $'\pi$ en $'\mathbf{P}$ zijn kortweg, want de diagonaalelementen zijn nul,

$${}'\pi = {}'\pi\, {}'\mathbf{P}$$

met $'\pi = ({}'\pi_{CPU}, {}'\pi_{I/O}, {}'\pi_{gebruiker})$ en $'\mathbf{P}$ overeenkomstig.

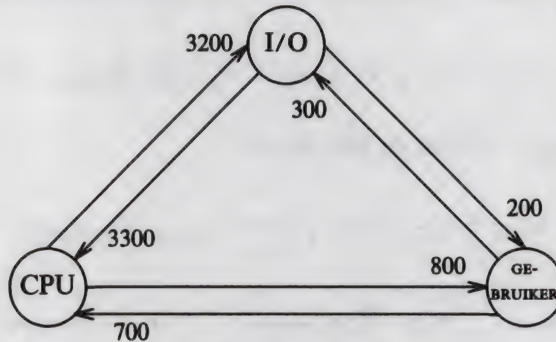
Deze operationele relatie heeft dezelfde vorm als de welbekende betrekking $\pi = \pi\mathbf{P}$. De meetinformatie in de *DOOR's* levert gemeten waarden voor zowel de schatters voor π als voor $\mathbf{P}$. Tussen deze waarden bestaat altijd de 'evenwichtsvergelijking' $'\pi = {}'\pi\, {}'\mathbf{P}$, tenminste als er wordt afgezien van randeffekten.

In de voorgaande hoofdstukken kwamen we bij de operationele aanpak steeds soortgelijke 'operationele' vergelijkingen tegen. Zullen en mogen we ook hier concluderen dat voor Markovketens altijd de relatie $\pi = \pi\mathbf{P}$ geldt voor de kansen op optreden en de overgangskansen?

11.2.2. voorbeeld: metingen accessen per usercommando

In een aaneengesloten meetduur van 2000 sekonden worden via de terminals 1000 usercommando's ingebracht. Uit de registratie van deze commando's blijkt het volgende. Er beginnen 700 commando's met een verwerking door de CPU. Het I/O-apparaat wordt in de meetduur 3500 maal aangedaan door een usercommando (3500 bezoeken bij I/O). De CPU verwerkt 4000 bezoeken. De verschillen tussen het totaal aantal overgangen naar een toestand en uit een toestand worden consequent verwaarloosd, alle 'randeffekten' zouden nul zijn.

Voor deze meting is $DOOR_{gebruiker\ CPU}$ = 700, er zijn 700 overgangen van gebruiker naar CPU (nieuwe opdrachten). $DOOR_{gebruiker\ I/O}$ = 1000−700 = 300. Daarmee zijn er 3500−300 = 3200 overgangen van CPU naar I/O: $DOOR_{CPU\ I/O}$ = 3200. Idem $DOOR_{CPU\ gebruiker}$ = 4000−3200 = 800. Maar ook $DOOR_{I/O\ gebruiker}$ = 1000−800 = 200 en $DOOR_{I/O\ CPU}$ = 3500−200 = 3300. De tellingen staan in figuur 11.1.



Figuur 11.1. Tellingen accessen per usercommando.

De gemeten waarde van de operationele schatter $'\pi_{CPU}$ is 4000/(1000+3500+4000) = 8/17.

De waarde voor de operationele schatter voor de vertakkingsverhouding bij de CPU is

$$'\mathbf{P}_{CPU\ I/O} = \frac{3200}{4000} = 4/5$$

In totaal geldt $'\pi = {}'\pi{}'\mathbf{P}$ met

$$'\pi = (8/17, 7/17, 2/17)$$

$$'\mathbf{P} = \begin{bmatrix} 0 & 4/5 & 1/5 \\ 33/35 & 0 & 2/35 \\ 7/10 & 3/10 & 0 \end{bmatrix}$$

Het gegeven 'de meetduur is 2000 sekonden' is helemaal niet gebruikt, dit natuurlijk omdat bij Markovketens de tijd 'stapsgewijs' verloopt. Daardoor zit alle relevante informatie in de aantallen, die in die tijdsduur gemeten zijn.

11.3. DRAAGWIJDTE OPERATIONELE RELATIE (FILOSOFISCH INTERMEZZO)

Uit de behoudswetten *IN* = *UIT* volgt een betrekking tussen de operationele schatters $'\pi$ voor de kansen op optreden en $'\mathbf{P}$ voor de overgangskansen. Deze betrekking $'\pi = '\pi'\mathbf{P}$ is van precies dezelfde vorm als de relatie $\pi = \pi\mathbf{P}$ voor ergodische Markovketens.

We vonden dit voor het eenvoudige voorbeeld van de usercommando's. De 'operationele' afleiding van de basisrelatie $\pi = \pi\mathbf{P}$ kan steeds gegeven worden als er overgangen zijn tussen een beperkt aantal toestanden (§11.4). Het is verrassend dat de relatie zo zonder omhalen opduikt in een operationele behandeling.

Een operationele betrekking is steeds in de eerste plaats een numerieke identiteit tussen meetgrootheden. Dat is ook hier het geval. Tussen de gemeten waarden voor de operationele schatters bestaat het aangegeven verband. Gelijkheden tussen operationele schatters worden gebruikt met de uiteindelijke bedoeling uitspraken te doen over 'echte' gemiddelde waarden en kansen. De problemen die zich bij zo'n extrapolatie voordoen, moeten hier duidelijk aan het licht komen, omdat de andere wegen om tot $\pi = \pi\mathbf{P}$ te komen al zijn onderzocht. En wat blijkt?

De operationele afleiding van de betrekking $'\pi = '\pi'\mathbf{P}$ tussen de operationele schatters geldt ook voor niet-ergodische Markovketens. De overeenkomstige betrekking $\pi = \pi\mathbf{P}$ voor de kansen op de lange termijn π geldt echter bij zulke ketens niet (§10.2.3). Over kansen op de lange termijn kan bij een niet-ergodische Markovketen zelfs vaak niet eens gesproken worden. Bij een absorberende Markovketen bijvoorbeeld komt het systeem op den duur in een van de absorberende toestanden. In de redeneringen voor operationele schatters wordt altijd aangenomen dat randeffekten van geen belang zijn, zulke verschillen worden consequent weggelaten in alle voorgaande 'operationele afleidingen' van relaties. Dit systematisch verwaarlozen van randeffekten is echter kennelijk niet altijd verdedigbaar.

De waarden voor de operationele schatters die in een meetduur van lengte *MEETDUUR* worden bepaald, leveren bij niet-ergodische Markovketens slechts uitspraken over het gemiddeld verloop in de tijd over tijdsduren in een aanloopfase (§9.6.1). DENNING en BUZEN hebben een filosofie rond hun operationele analyse ontwikkeld, waarin ze zich met deze uitspraken tevreden stellen, zulke uitspraken zouden voldoende representatief zijn.

Aan het 'tegenvoorbeeld' van de niet-ergodische Markovketens zien we dus duidelijk de complicaties van de operationele aanpak. De veronderstellingen behoren te worden aangevuld! Als we het verband tussen operationele schatters willen gebruiken om tot uitspraken over de stationaire fase te komen, moeten we eerst —op andere gronden dan deze operationele betrekking— zeker weten dat er een evenwichtstoestand bestaat.

Het tegenvoorbeeld uit het hoofdstuk INLEIDING OPERATIONELE ANALYSE (opgave 3.1) is al een eerste waarschuwing in deze richting.

Daarmee zijn we er nog niet. Ook als er een stationaire fase bestaat, zoals bij ergodische ketens, zijn er verdere vraagtekens. In de operationele aanpak wordt

geen enkele uitspraak gedaan over de betekenis van de overgangskansen. Hangt de overgangskans uitsluitend van de huidige toestand af of ook van voorgaande toestanden? In de operationele aanpak wordt simpelweg gemiddeld over alle voorgeschiedenissen die zich hebben voorgedaan, de voorgeschiedenis bij een overgang wordt niet bijgehouden. De overgangskansen kunnen van de k-de orde zijn, met k eindig. Maar wat k is, daarover wordt geen informatie verkregen.

Het is in zo'n geval het verstandigst de eenvoudigste aanname te maken en te doen† alsof $k=1$. Als deze aanname wordt gemaakt worden $'\pi$ en $'\mathbf{P}$ de operationele schatters voor respectievelijk de kansen op optreden in de stationaire fase en de overgangsmatrix van een ergodische Markovketen van de eerste orde. Daarna kan uit de relatie tussen de operationele schatters, via de bij de overgang van een beperkte op een uitgebreide vorm steeds gevolgde redenering, worden besloten dat voor de 'eigenlijke' kansen (de kansen horend bij deze ergodische Markovketen van de eerste orde) de relatie $\pi=\pi\mathbf{P}$ moet gelden. En dat is correct. We hebben dit met een heel andere redenering vastgesteld bij de analyse van kansen op de lange termijn in het hoofdstuk ERGODISCHE MARKOVKETENS.

Als de echte overgangsmatrix ergodisch van de k-de orde is, loopt de redenering als volgt. (De echte overgangsmatrix tussen de huidige toestand plus de voorgeschiedenis van $(k-1)$ stappen en de komende toestand plus de voorgeschiedenis van $(k-1)$ stappen blijft onbekend, waarnemingsmateriaal dat bestaat uit de aantallen overgangen tussen de huidige toestand en de komende toestand (de *DOOR*'s) staat niet toe daarover informatie te geven.)

Kansen op de lange termijn verschillen per voorgeschiedenis. Wanneer in het voorbeeld de echte overgangsmatrix van de derde orde is, verschilt de kans op de lange termijn op de sequentie 'bij CPU, bij I/O, bij gebruiker' van de kans op de sequentie 'bij I/O, bij CPU, bij gebruiker'. Middelen over alle voorgeschiedenissen van twee stappen, die aan de toestand 'gebruiker' vooraf kunnen gaan, levert de 'overall' kans op optreden van 'bij de gebruiker'. Door dit middelen ontstaan algemeen de overall kansen op optreden π.

Er kan daarnaast over de overgangskansen uit de overgangsmatrix van de k-de orde worden gemiddeld, waardoor een 'overall' 1-staps overgangskans ontstaat, en daarmee een 'overall' eerste orde overgangsmatrix $\mathbf{P}$.

Ook als de echte overgangsmatrix van de k-de orde is, bestaan er dus overall kansen op optreden en een overall overgangsmatrix van de eerste orde. De operationele schatters $'\pi$ en $'\mathbf{P}$ schatten deze overall kansen op optreden, respectievelijk deze overall overgangsmatrix! Er mag voor de overall kansen in de stationaire fase en de overall overgangsmatrix worden geconcludeerd dat er een betrekking $\pi=\pi\mathbf{P}$ geldt.

In het voorbeeld over overgangen tussen UNIX-filesystemen uit §10.4.2 berekenden we de overall overgangsmatrix (de eerste-orde overgangsmatrix) uit de

---

† Binnen de informatietheorie kan hiervoor een beroep worden gedaan op het principe van 'maximalisatie van entropie'.

overgangsmatrix van hogere orde. We losten eerst voor de gedetailleerde hogere orde matrix de vergelijking $\pi = \pi\mathbf{P}$ op en vonden daarmee de kansen op de lange termijn op de sequenties. Deze kansen samen met de hogere-orde overgangsmatrix maakten het mogelijk de overall overgangsmatrix van de eerste orde op te stellen en de overall kansen op optreden uit te rekenen. We zagen daar dus hoe de middeling binnen het stochastische kader concreet verloopt.

De operationele aanpak geeft duidelijk aan dat een Markovketen van hogere orde zich voor de overall kansen steeds laat herleiden tot een eerste orde Markovketens. De effektieve overgangskansen uit de 'overall' eerste orde keten worden door middeling bepaald uit de k-de orde overgangsmatrix.

Het blijft dus oppassen. Toch zijn de opgeworpen bedenkingen niet typisch voor de operationele aanpak, ze doen zich eigenlijk steeds voor als men vanuit meetresultaten tot uitspraken over 'echte' grootheden wil komen. Heel vaak laat het meetmateriaal pas interessante uitspraken toe als er eerst enkele basisveronderstellingen over het te meten object zijn gemaakt. We plaatsten al eerder kanttekeningen van deze aard, bijvoorbeeld in het hoofdstuk ERGODISCHE MARKOVKETENS in §9.7 over het pendelend systeemproces 'progressor'.

De operationele weg geeft snel en direkt een uitzicht op de uiteindelijk te behalen oogst, maar om die binnen te halen moet met overleg te werk worden gegaan.

11.4. BINNENDRUPPELEN EN WEGLEKKEN

De relatie $\pi = \pi\mathbf{P}$ is dus welhaast 'alom tegenwoordig', het is een echte 'basisrelatie'. Dit komt ook omdat de betrekking uitdrukking geeft aan wat nu eigenlijk een stationaire fase is en hoe daarin de overgangskansen en de kansen op optreden worden bepaald. Aan het eind van al onze —zo langzamerhand wel wat talrijke— beschouwingen over Markovketens gaan we de relatie $\pi = \pi\mathbf{P}$ daarom nog eens speciaal met het oog hierop bekijken. Daarbij wordt zowaar nog iets nieuws ontdekt: een heel algemene manier om de relatie in alle voorkomende gevallen snel op te sporen, zonder eerst de overgangsmatrix uit te schrijven en dan de vermenigvuldiging van rijvector en matrix te plegen. We gebruiken deze laatste paragraaf over $\pi = \pi\mathbf{P}$ ook om alles nog eens in algemene termen samen te vatten.

We gaan er van uit dat alleen overgangen tussen discrete toestanden worden bijgehouden. In de loop van de tijd gaat het systeem van toestand naar toestand. In de meetduur *MEETDUUR* wordt gemeten:

- Het aantal overgangen $DOOR_{ij}$ van toestand_i naar toestand_j voor alle i en j.

In de meetduur is UIT_i het totale vertrek uit toestand_i: het aantal overgangen van toestand_i naar andere toestanden.

$$UIT_i = \sum_j DOOR_{ij}$$

In die tijd is IN_i de totale binnenkomst in toestand$_i$: het aantal overgangen vanuit andere toestanden naar toestand$_i$.

$$IN_i = \sum_j DOOR_{ji}$$

De sommaties over j zullen vaak i zelf ook omvatten, in veel Markovketens zijn er immers overgangen van een toestand naar zichzelf mogelijk.

Het is kenmerkend voor een systeem in de evenwichtsfase dat enerzijds de toestand van het systeem telkens verandert en anderzijds de mix aan toestanden gemiddeld genomen dezelfde blijft. De keren 'vertrek uit toestand$_i$' worden gemiddeld gecompenseerd door de keren 'binnenkomst in toestand$_i$'. Dat houdt als 'behoudswet' in dat $IN_i \approx UIT_i$ of onder verwaarlozing van de randeffekten

$$\sum_j DOOR_{ji} = \sum_j DOOR_{ij}$$

De operationele schatter voor de overgangskans van i naar j is

$$'\mathbf{P}_{ij} = DOOR_{ij} / UIT_i$$

en de operationele schatter $'\pi_i$ voor de kans op optreden van toestand$_i$ is (met $UIT = \sum_i UIT_i$)

$$'\pi_i = UIT_i / UIT$$

Deze operationele schatters worden in de behoudswet $IN_i = UIT_i$ ingevuld. Daarvoor wordt de behoudsrelatie tussen de *DOOR*'s herschreven tot (mits UIT_j, UIT_i niet 0)

$$\sum_j (UIT_j / UIT)(DOOR_{ji} / UIT_j) = (UIT_i / UIT) \sum_j DOOR_{ij} / UIT_i$$

Dat is met operationele schatters

$$\sum_j {'\pi_j} {'\mathbf{P}_{ji}} = {'\pi_i} \sum_j {'\mathbf{P}_{ij}}$$

In deze sommen kunnen we de termen met $i = j$ wel uitsluiten, die zijn toch links en rechts gelijk. Gesommeerd over alle j is $\sum_j {'\mathbf{P}_{ij}} = 1$.

Er komt dan

$$\sum_{j \neq i} {}'\pi_j \, {}'\mathbf{P}_{ji} = {}'\pi_i (1 - {}'\mathbf{P}_{ii})$$

Zo is algemeen de afleiding herhaald, die we zojuist in §11.2.1 voor drie toestanden gaven, met toestand_1 = CPU, toestand_2 = I/O en toestand_3 = gebruiker.

De verkregen evenwichtsrelatie is natuurlijk niets anders dan de basisrelatie $'\pi = '\pi\,'\mathbf{P}$ in een lichtelijk herschreven vorm. Maar deze verbogen vorm laat een interessante interpretatie toe.

We letten op de drie voorgaande vormen van de evenwichtsvergelijking. De *rechterkant* van de vergelijking ontstaat uit het totale *vertrek* uit toestand_i, zoals in de vorm met de *DOOR*'s staat aangegeven. Door te delen door *UIT* komt er aan de rechterkant het gemiddelde vertrek per toestandswisseling. Dit blijkt te schrijven te zijn als het produkt van twee termen: de relatieve frequentie van voorkomen ($'\pi_i$, corresponderend met π_i) van toestand_i maal de totale kans (som van $'\mathbf{P}_{ij}$, respectievelijk $\mathbf{P}_{ij}$) om vanuit toestand_i naar andere toestanden te vertrekken. Of, wat anders gezegd, de kans om bij een toestandsovergang de toestand_i te verlaten is het produkt van de kans op optreden van toestand_i maal de som van de overgangskansen vanuit toestand_i.

Deze rechterkant van de evenwichtsvergelijking kan plastisch worden betiteld met *weglekken*. In het rechterlid staat dat de kans op weglekken uit toestand_i gelijk is aan de kans dat toestand_i optreedt maal de totale overgangskans om van toestand_i naar een andere toestand te gaan.

De *linkerkant* van de evenwichtsvergelijking ontstaat uit de totale *binnenkomst* in toestand_i (linkerkant van de vorm met de *DOOR*'s). Door te delen door *UIT* komt er de gemiddelde binnenkomst per toestandswisseling. Dit blijkt te schrijven te zijn als een som van produkten van twee termen: de relatieve frequentie van voorkomen $'\pi_j$ van een toestand_j maal de overgangskans $'\mathbf{P}_{ji}$ om vanuit die toestand naar toestand_i te gaan.

Wat de linkerkant aangeeft betitelen we als *binnendruppelen*; de linkerkant van de evenwichtsrelatie sommeert het binnendruppelen. Er staat in de linkerkant van de evenwichtsvergelijking dat de kans op binnendruppelen in toestand_i gelijk is de kans dat een andere toestand dan toestand_i optreedt, maal de kans om vanuit die toestand naar toestand_i te verhuizen en dat gesommeerd over alle mogelijkheden.

Zo blijkt dat de 'echte' evenwichtsrelatie $\pi = \pi\mathbf{P}$ ook ontstaat door het weglekken uit een toestand gelijk te stellen aan het binnendruppelen in die toestand! De kans op 'binnendruppelen', het linkerlid, is gelijk aan de kans op 'weglekken', het rechterlid. We noemen dit het principe van *'binnendruppelen is weglekken'*. In de stationaire toestand veranderen de toestanden, maar de kansen op de toestanden blijven gelijk, de gemiddelde 'bezetting' van een toestand verandert niet; er is gemiddeld genomen nòch een emigratieoverschot, nòch een immigratieoverschot. Dit kan alleen als er compensaties optreden, die met het *evenwichtsprincipe* van

binnendruppelen is weglekken beschreven worden. De vergelijkingen

$$\sum_{j \neq i} \pi_j \mathbf{P}_{ji} = \pi_i (1 - \mathbf{P}_{ii})$$

karakteriseren dus de *evenwichtstoestand*.

Vergelijkingen, die uit dit principe volgen, worden met recht altijd *evenwichtsvergelijkingen* (balance equations) genoemd. Zulke vergelijkingen worden met het evenwichtsprincipe op een vanzelfsprekende manier rechtstreeks uit de overgangskansen afgeleid.

De belangrijke *Markovprocessen* zijn een generalisatie van Markovketens naar een continu tijdsverloop (§16.2). Voor deze algemene processen worden de evenwichtsvergelijkingen ook met het principe van 'binnendruppelen is weglekken' afgeleid. In §16.2 worden bijvoorbeeld de evenwichtsvergelijkingen voor het M/M/1 systeem opgesteld. Het evenwichtsprincipe is een algemene bouwsteen voor berekeningen in de *stationaire fase*.

11.5. BEPALING EVENWICHTSVERGELIJKING BIJ MARKOVKETENS

Door het evenwichtsprincipe 'binnendruppelen is weglekken' toe te passen worden de evenwichtsvergelijkingen zonder meer vanuit de specificaties opgeschreven. De weg via het opstellen van de overgangsmatrix en het uitschrijven van $\pi = \pi\mathbf{P}$ is kortgesloten.

Er wordt bij het vinden van de evenwichtsvergelijkingen vaak gebruik gemaakt van diagrammen waarin alle toestanden, met de kansen op de overgangen er tussen, staan aangegeven. Deze kunnen naar binnendruppelen en weglekken worden gegroepeerd. Het opstellen van de evenwichtsvergelijkingen gebeurt aan de hand van deze plaatjes, die *toestandsdiagrammen* (state diagrams) worden genoemd. Voor alle toestanden wordt successievelijk 'binnendruppelen is weglekken' toegepast, waardoor elke toestand een evenwichtsvergelijking levert.

Figuur 11.2 en figuur 11.3 zijn de toestandsdiagrammen voor twee voorbeelden uit het hoofdstuk ERGODISCHE MARKOVKETENS. Ook figuur 9.4 uit §9.9 is een toestandsdiagram.

11.5.1. voorbeelden

In het standaardvoorbeeld (§9.2) met de schutters worden maar twee toestanden onderscheiden, 'raak' en 'mis'. De evenwichtsvergelijking voor de toestand 'raak' vinden we met het evenwichtsprincipe als volgt.

De kans om een overgang uit de toestand 'raak' te maken is $\pi(0)$ (de kans dat raak de beurt heeft) maal de kans $\mathbf{P}_{01}$ om vanuit raak mis te schieten. Deze laatste kans is 1/5, zodat het 'weglek' deel van de relatie gelijk aan $\pi(0) \times 1/5$.

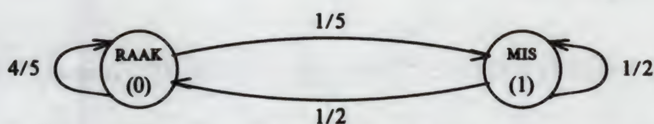
De kans om een overgang naar 'raak' te maken is $\pi(1)$ (de kans dat mis de beurt heeft) maal de overgangskans van 1/2 van mis naar raak. Dit is het 'binnendruppel' deel van de relatie.

De evenwichtsrelatie die bij de toestand 'raak' hoort, is dus

$$1/2\,\pi(1) \;=\; 1/5\,\pi(0)$$

Inderdaad een van de vergelijkingen, die we uit $\pi = \pi\mathbf{P}$ bij de bepaling van de stationaire toestand in §9.5.2 destilleerden.

De vergelijking kan aan de hand van figuur 11.2 worden opgesteld. Kijk naar de ingaande en de uitgaande pijlen voor de knoop 'raak', ingaand correspondeert met binnendruppelen en uitgaand met weglekken.



Figuur 11.2. Binnendruppelen en weglekken in het standaardvoorbeeld.

Op dezelfde manier vinden we de evenwichtsvergelijking voor de toestand 'mis'. Het binnendruppelen in toestand 'mis' levert $\pi(0)\times 1/5$: de kans op raak, maal de overgangskans van raak op mis. Het weglekken uit toestand 'mis' geeft $\pi(1)\times 1/2$. Omdat binnendruppelen en weglekken elkaar in de evenwichtstoestand compenseren, geldt

$$1/5\,\pi(0) \;=\; 1/2\,\pi(1)$$

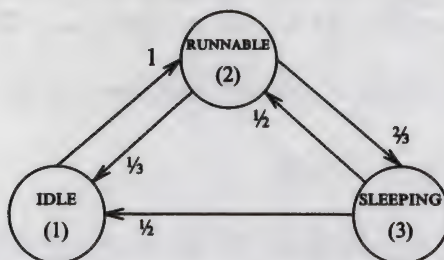
Dat is weer dezelfde vergelijking, de evenwichtsvergelijkingen zijn afhankelijk.

Precies zo gaat het bij het voorbeeld uit §9.7 over de 'Progressor' (de gegevens zijn overgenomen in figuur 11.3). De evenwichtsvergelijking voor de toestand 'idle' volgt door de kans op binnendruppelen van $(\pi_2 \times 1/3 + \pi_3 \times 1/2)$ gelijk te stellen aan de kans op weglekken van $\pi_1 \times 1$ (de nummering van de toestanden is idle, runnable, sleeping). Dus

$$1/3\,\pi_2 \;+\; 1/2\,\pi_3 \;=\; \pi_1$$

Deze vergelijking vonden we in §9.7 door eerst de overgangsmatrix op te stellen.

In figuur 9.4 uit §9.9 worden voor het voorbeeld met de usercommando's alle overgangskansen aangegeven. Ook deze figuur is een toestandsdiagram, aan de hand van dit plaatje zijn de evenwichtsvergelijkingen direkt op te schrijven. Voor



Figuur 11.3. Binnendruppelen en weglekken voor voorbeeld 'Progressor'. Geen overgangen van een toestand naar dezelfde toestand.

de toestand 'bij de CPU' geeft het evenwichtsprincipe van binnendruppelen (links) is weglekken (rechts), dat

$$\pi_{paging} + \pi_{I/O-1} + \pi_{I/O-2} + \tfrac{1}{2}\pi_{I/O-3} + \tfrac{1}{2}\pi_{I/O-4} + \pi_{gebruiker} = \pi_{CPU}$$

En voor I/O-4

$$1/36\,\pi_{CPU} = \pi_{I/O-4}$$

Dit zijn twee van de vergelijkingen uit $\pi = \pi\mathbf{P}$, die in §9.9.1 (na figuur 9.5) zijn opgelost.

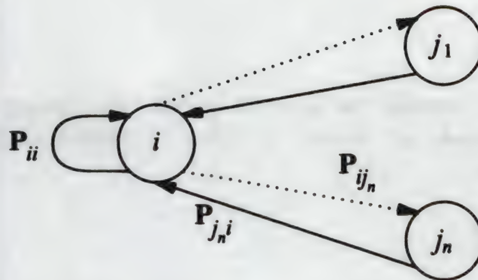
11.5.2. algemeen toestandsdiagram

De evenwichtsvergelijking

$$\sum_{j \neq i} \pi_j \mathbf{P}_{ji} = \pi_i (1 - \mathbf{P}_{ii})$$

kan altijd op een toestandsdiagram worden afgebeeld. Voor de evenwichtsvergelijking voor toestand$_i$ is alleen wat in figuur 11.4 staat van belang. Bij de ingaande pijlen horen de bijdragen van binnendruppelen uit het linkerlid. Met de uitgaande pijlen corresponderen de bijdragen door weglekken uit het rechterlid. De totale kans op weglekken uit een toestand is gelijk aan de kans op optreden van de toestand, vermenigvuldigd met de totale kans op een overgang naar elders. Deze laatste kans is 1, verminderd met de kans op een overgang naar zichzelf. Gelijkstellen van de totale bijdragen van binnendruppelen en weglekken levert de evenwichtsvergelijking voor de toestand.

Voor een continu verloop in de tijd worden de evenwichtsvergelijkingen met soortgelijke toestandsdiagrammen gevonden (figuur 16.1-2 uit §16.2).



Figuur 11.4. Evenwichtsvergelijking voor toestand_i. Tijdsverloop stapsgewijs. Ingaande pijlen: binnendruppelen in toestand_i. Uitgaande pijlen: weglekken uit toestand_i. Alleen de overgangen van en naar toestand_i zijn getekend. De kans op binnendruppelen vanuit toestand_{j_k} is $\pi_{j_k}\ \mathbf{P}_{j_k i}$. Voor het binnendruppelen in toestand_i moet over alle andere toestanden (j_1 tot en met j_n) worden gesommeerd. De kans op weglekken uit toestand_i naar toestand_{j_k} is $\pi_i\ \mathbf{P}_{ij_k}$. De totale overgangskans bij weglekken is 1, verminderd met de kans $\mathbf{P}_{ii}$ op een overgang terug naar toestand_i.

11.6. SAMENVATTING

De relatie tussen de operationele schatters voor de kansen op optreden en de overgangskansen

$$'\pi = '\pi'\mathbf{P}$$

kan worden generaliseerd tot de evenwichtsvergelijking

$$\pi = \pi\mathbf{P}$$

Dit is alleen mogelijk in de stationaire fase. De operationele relatie maakt duidelijk dat deze evenwichtsvergelijkingen voor de stationaire toestand direkt gevonden kunnen worden met het principe van 'binnendruppelen is weglekken'. De handigste vorm is (deze geldt voor systemen met een eindig aantal verschillende toestanden)

$$\sum_{j \neq i} \pi_j \mathbf{P}_{ji} = \pi_i (1 - \mathbf{P}_{ii})$$

Deze betrekking zegt dat voor elke toestand in de stationaire fase het weglekken uit de toestand (rechterlid) en het binnendruppelen in de toestand (linkerlid) elkaar in evenwicht houden. Door dit evenwichtsprincipe achtereenvolgens voor alle toestanden toe te passen worden de evenwichtsvergelijkingen gevonden zonder dat eerst de overgangsmatrix moet worden opgesteld.

11.7. OPGAVE

opgave 11.1: evenwicht

Geef voor enkele voorbeelden en opgaven uit de beide voorgaande hoofdstukken de toestandsdiagrammen. Stel daarna de evenwichtsvergelijkingen op aan de hand van deze diagrammen.

12

Signalen en Poissonproces

12.1. INLEIDING

Het 'volgen' van wat zich in de tijd afspeelt heeft heel wat kanten. We gaan nu letten op 'referentiepunten', die in de tijd gezet worden door —zeg maar, vrijblijvend— een of ander proces; we vragen ons af hoeveel referentiepunten er gemiddeld gezet worden en wat de verdeling is van de tijdsduren er tussen. Men zegt vaak dat we het optreden bijhouden van *events*, dat zijn momenten waarop de een of andere wijziging te registreren valt; als men 'Nederlands' wil gebruiken heeft men het over *'gebeurtenissen'*.

Wat als 'events' worden beschouwd hangt van de interesse van de waarnemer af. Heel vaak heeft deze alleen oog voor bepaalde toestandswijzigingen, een event is dan het optreden van zo'n wijziging. Heel algemeen gesproken is elk 'voorval' een event, het voorval bepaalt een tijdsmoment waaraan gerefereerd kan worden. Het binnenkomen van een interrupt of het gevuld raken van een bufferplaats kan bijvoorbeeld een event zijn.

Het optreden van een event wordt ook wel aangeduid als het optreden van een *signaal*. Omdat dit misschien het neutraalste en minst belaste† woord is, zullen we in het volgende niet meer van events maar van signalen spreken. Signalen treden op, komen binnen of hoe men het maar zien en zeggen wil.

Ons probleem is eerst om algemeen te beschrijven hoeveel signalen er gemiddeld optreden en hoelang gemiddeld de tussenpozen tussen signalen zijn (een interval tussen twee opvolgende signalen noemen we, als in het hoofdstuk RESTDUREN EN WACHTDUREN, een *tussenpoos*; in de literatuur heet het een *interarrival time*).

---

† Misschien is het toch goed er op te wijzen dat het begrip signaal veel ruimer is dan het beperkte 'hardware-signaal'.

Een gebruikelijke dagelijkse maat voor het optreden van signalen is 'zoveel per tijdseenheid', of gemiddeld zoveel in een tijdinterval van die en die duur. Over de betekenis hiervan gaan we eerst wat filosoferen. Het gemiddeld aantal signalen in een *willekeurig* tijdinterval van lengte d blijkt gelijk te zijn aan λd, als λ het aantal signalen per tijdseenheid is en d de duur van het tijdinterval. Meer informatie is er niet te halen uit het gegeven dat de *snelheid van optreden* λ is.

Als echter bovendien bekend is dat de signalen op *lukrake momenten* komen kan ook de verdeling rond het gemiddelde worden aangegeven, en dat voor 'elk' interval. Signalen die op lukrake momenten optreden, zijn signalen volgens een *Poissonproces*. De aantallen signalen in een interval van lengte d zijn bij een Poissonproces verdeeld volgens de *Poissonverdeling* met als verwachtingswaarde λd; deze Poissonverdeling is bekend uit de statistiek. De tussenpozen tussen signalen hebben een negatief exponentiële verdeling met als verwachtingswaarde $1/\lambda$.

Het is kenmerkend voor een Poissonproces dat niet alleen in een 'lukraak' interval, maar in 'elk' interval van lengte d het gemiddelde aantal signalen λd is. Deze eigenschap zullen we de 'basiseigenschap' van het Poissonproces noemen, uitgaande van deze eigenschap worden alle verdere karakteristieken van dit proces afgeleid.

Bij en in een computerconfiguratie gebeurt —globaal gezien— veel lukraak en op willekeurige momenten. Het Poissonproces is dan ook van veel belang bij de analyse van computer-prestaties. In dit hoofdstuk onderzoeken we de eigenschappen van dit 'volmaakt lukrake' proces.

12.2. GEMIDDELD PER TIJDINTERVAL

12.2.1. betekenis per

Veronderstel dat er bekend is, dat per uur gemiddeld 600 signalen optreden. Hoeveel signalen treden er dan gemiddeld per minuut op? Het voor de hand liggende en correcte antwoord is 10. Hoeveel per sekonde? 1/6. Hoeveel treden er gemiddeld in één millisekonde op? Antwoord is 1/6000.

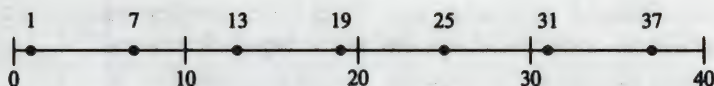
Dit laatste roept misschien wel wat vragen op. Kunnen we op een tijdschaal in millisekonden, die zo heel anders is dan die, waarin de oorspronkelijke mededeling gedaan wordt, nog wel handhaven dat het gemiddeld aantal signalen in z sekonden gevonden wordt als

$$\textit{gemiddeld aantal in z sekonden} = z \times 600/3600 = z/6$$

voor iedere waarde van z?

Inzicht is vaak te verkrijgen door een extreem geval onder ogen te zien. Extreem is hier dat signalen met vaste tussenpozen optreden, zonder een spreiding in de duur van de tussenpoos. Dit gaan we daarom eerst bekijken.

Denk eens in dat de signalen precies om de zes sekonden optreden en dat het registreren begint op tijdstip 0 (zie figuur 12.1).



Figuur 12.1. Signalen met vaste tussenpozen van 6 sekonden, eerste signaal 1 sekonde na tijdstip 0. Een signaal is aangegeven met een •

Als geconstateerd is dat op het tijdstip $t = 1$ een signaal kwam, komt het volgende op tijdstip 7, daarna op 13, 19, 25, 31, enzovoort. In de eerste tien sekonden dus 2 signalen op de tijdstippen 1 en 7, in de tweede tien sekonden idem 2 signalen op de tijdstippen 13 en 19, in de derde tien sekonden echter maar één op tijdstip 25. Het vierde interval van tien sekonden is een replica van het eerste, dus weer de 2 signalen; in de volgende intervallen van tien sekonden weer 2 en 1 en opnieuw 2, 2, 1 signalen, enzovoort. Er zijn dus intervallen van tien sekonden, waarin twee en intervallen van tien sekonden waarin maar één signaal optreedt en deze intervallen volgen elkaar op in een vast patroon met als aantallen signalen 2, 2, 1.

Het aantal signalen dat in het eerste interval van tien sekonden van zo'n sequentie optreedt is 2 en daarmee ook gemiddeld 2. Zou men zich beperken tot zulke intervallen, dan is het aantal signalen in tien sekonden dus gemiddeld 2. Dat is het ook als men alleen let op de eerste en de tweede intervallen. Voor de derde intervallen uit de aangegeven sequentie van driemaal tien sekonden is het aantal signalen echter gemiddeld 1.

Toch is deze informatie niet wat bedoeld zal worden als in het gewone spraakgebruik simpelweg gevraagd wordt naar het gemiddeld aantal signalen per 10 sekonden. Er wordt via het woordje *per* geïnformeerd naar het gemiddeld aantal over een interval van lengte 10 sekonden, waarbij duidelijk niet wordt aangegeven over welk interval —of 'type' interval— van lengte 10 het gaat. Er zal, bewust of onbewust, bedoeld zijn dat de intervallen, waarover gemiddeld wordt, in geen enkel opzicht speciaal zijn. Met *per* interval van lengte 10 wordt kennelijk gemeend: over een lukraak (random) gekozen interval van lengte 10.

We behoren ons dus niet te beperken tot speciale intervallen, zoals de typen uit de beschreven indeling. Er zal stellig gemiddeld moeten worden over alle typen. De aangegeven typen komen even vaak voor, de kans op optreden is 1/3. Middeling over deze typen zou opleveren

$$\textit{gemiddeld aantal in } 10 \textit{ sekonden} = (2+2+1)/3 = 5/3$$

En als we de intervallen wat anders leggen en bijvoorbeeld het eerste interval laten lopen van 4 tot 14, het tweede van 14 tot 24 en het derde van 24 tot 34, komen we dan ook op dit zelfde uit? Gelukkig wel.

We moeten de intervallen echter echt lukraak kiezen, dus zo maar een beginpunt nemen en het aantal signalen in de dan volgende 10 sekonden bepalen. Er blijkt nu dat er een getal wordt gevonden vlak bij 5/3 als er

- heel veel lukrake intervallen van 10 sekonden worden bekeken en het totale aantal daarin optredende signalen wordt bepaald, waarna
- dat aantal gedeeld wordt door het aantal intervallen.

Hier staat —zegt men in de omgangstaal— dat er in een interval van 10 sekonden gemiddeld 5/3 signalen komen. Of (preciezer) dat de verwachtingswaarde van het aantal signalen in 10 sekonden 5/3 is. De bepaling 'lukraak' ontbreekt!

Alledaagse taal is heel subtiel, het weglaten van een nadere specificatie betekent meestal een impliciete middeling over alle mogelijkheden. Hier gaat de middeling over de oneindig veel lukrake mogelijkheden. Bij de invoering van de operationele analyse liepen we al tegen deze eigenaardigheid van de gewone taal aan, zie bijvoorbeeld het hoofdstuk RESPONSETIJDRELATIES.

Wanneer de tussenpozen niet steeds exact 6 sekonden zijn ligt het probleem niet anders. Ook dan is gemiddeld over lukrake intervallen van duur z

$$\textit{gemiddeld aantal in } z \textit{ sekonden} \;=\; z/6 \textit{ signalen}$$

Of kortweg: de verwachtingswaarde van het aantal signalen in z sekonden is $z/6$ signalen. Het is deze informatie die gewoonlijk kort wordt weergegeven als *gemiddeld 1/6 signaal per sekonde*, in een uitdrukking met *per*.

De algemene vraag naar het gemiddelde aantal signalen over een interval van een bepaalde tijdsduur ('per dit interval') moet dus geïnterpreteerd worden als een vraag naar het gemiddeld aantal signalen over een lukraak gekozen interval van deze lengte. Dat is dus volgens de visie van de lukrake waarnemer!

Er zijn echt geen problemen bij schaalverandering; er treedt in een heel kort tijdinterval alleen zelden een signaal op, slechts in een gering aantal van de lukraak te kiezen intervallen treedt er een signaal op.

De algemene conclusie is weer te geven in een alledaagse vorm: wanneer er per tijdseenheid gemiddeld λ signalen optreden, zullen er per tijdinterval van lengte d gemiddeld λd signalen optreden. Een weinig verrassende uitspraak misschien, maar er moet ooit over nagedacht zijn. We noemen λ de *snelheid* (rate) waarmee signalen optreden, of de *snelheid van optreden*, binnenkomen, aankomst, enzovoort. In de literatuur wordt λ ook wel de *intensiteit* genoemd. De grootheid λ wordt uitgedrukt in signalen per tijdseenheid.

In een formulering met stochasten zegt men: In een interval treedt een aantal signalen op (er is een aantal voorvallen). Dit aantal noemen we kortweg het aantal signalen in dit tijdinterval. De stochast $N(d)$ gaat het aantal signalen in een

lukraak genomen tijdinterval van lengte d aangeven. Voor $N(d)$ zal gelden

$$E(N(d)) = \lambda d$$

Met bijvoorbeeld $\lambda = 1/6$ en $d = 10$ is

$$E(N(10)) = 5/3 \text{ signalen}$$

De kans op zeg 3 signalen in een lukraak tijdinterval van lengte 10 geven we aan als $Pr(N(10)=3)$.

12.2.2. voorbeeld: onderhoudsmonteur

Als een onderhoudstechnicus gemiddeld om de drie werkdagen de computer nakijkt, komt hij gemiddeld om de twee weken één keer en anders tweemaal in de week. Hij komt in een werkweek van 5 dagen gemiddeld 5/3 maal; $\lambda = 1/3$ en $d = 5$ met $E(N(5))=1/3\times5$. Hij komt ook gemiddeld 1/3 maal per dag. En dat alles wisten we natuurlijk al vanaf de tijd dat we met taal leerden om te gaan.

12.2.3. kans op een signaal

In een willekeurig tijdinterval van lengte d treden, zo stelden we vast, gemiddeld λd signalen op als er per tijdseenheid gemiddeld λ signalen optreden. Wanneer d heel klein is, wordt ook λd erg klein en zal het heel weinig voorkomen dat er in het intervalletje van lengte d een signaal optreedt. Nog zeldzamer zal de gebeurtenis zijn, dat er in het zeer korte intervalletje niet één, maar tweemaal een signaal optreedt. Dit laatste geldt zolang de signalen niet paarsgewijs strikt gesynchroniseerd optreden, maar zo'n speciaal geval is niet aan de orde; er wordt aangenomen dat signalen alleen toevallig tegelijkertijd optreden. We kunnen het zeer zeldzame optreden van meer dan één signaal in een zelfde tijdintervalletje rustig vergeten, mits we de lengte d van het intervalletje maar kort genoeg nemen, 'dat wil zeggen' infinitesimaal kort van lengte dt.

We willen dit preciezer formuleren. Als d erg klein is (dt) mogen we de kans op twee signalen in een tijdintervalletje van lengte dt verwaarlozen, dus in de notatie uit het slot van §12.2.1

$$Pr(N(dt)\geqslant 2) \approx 0$$

We stelden daar vast dat het gemiddelde aantal signalen in het interval van lengte dt gelijk is aan λdt, hoe kort dt ook is, of geformuleerd met de verwachtingswaarde $E(N(d))$ van $N(d)$

$$E(N(dt)) = \lambda dt$$

Uitschrijven van de verwachtingswaarde geeft

$$E(N(dt)) = 0 \times Pr(N(dt)=0) + 1 \times Pr(N(dt)=1) + 2 \times Pr(N(dt)=2) + \cdots$$

In het rechterlid telt de eerste term niet, want die is nul en ook niet de derde en volgende termen, die zijn omdat signalen alleen toevallig synchroon optreden vrijwel nul; dus alleen $Pr(N(dt)=1)$ resteert. Het linkerlid is λdt, zodat we overhouden

$$Pr(N(dt)=1) = \lambda dt$$

Voor een willekeurig tijdinterval van zeer korte lengte is dus *de kans op een signaal* in dat tijdinterval *evenredig met de lengte van het interval*, met als evenredigheidsfactor de 'snelheid van optreden' λ. De kans op een signaal is λdt, op geen signaal $1-\lambda dt$. Bij gemiddeld 1/6 signaal per sekonde is de kans op een signaal in 1 msek. —afgezien van hogere orde correcties— gelijk aan 1 op 6000.

Uitgaande van de snelheid van optreden, die het gemiddeld aantal signalen in een lukraak interval van elke mogelijke lengte bepaalt, zijn we door maar door te redeneren terecht gekomen op de kans op een signaal in een lukraak kort interval en zijn we dus van verwachtingswaarde naar kans gewandeld.

Voor elk patroon van signalen is dus de kans op een signaal in een lukraak heel kort tijdinterval bekend. Aanstonds zullen we proberen de kans op een signaal te vinden voor een lukraak interval dat niet erg kort is, maar 'van eindige lengte'.

12.2.4. de twee registraties van signalen

Optredende signalen kunnen altijd op twee manieren geregistreerd worden.

De eerste manier is: observeer gedurende de meetduur en noteer het *aantal* signalen dat optreedt, dus tel de momenten waarop signalen komen.

De tweede manier is: observeer gedurende de meetduur en noteer de tijdsduren tussen opeenvolgende signalen (de *tussenpozen*). De eerste registratie is uiteraard (afgezien van randeffekten) uit de tweede af te leiden.

Bij de eerste registratiemethode kunnen we rechtstreeks aflezen hoeveel signalen er in een bepaalde tijdsduur optreden, het is slechts een kwestie van blijven tellen. De stochast, waarvan we op deze manier de waarden noteren, is het aantal signalen in de meetduur d, zeg maar $M(d)$ of, als de huidige meetduur wordt gezien als een specimen van een willekeurige meetduur van lengte d: $N(d)$. Als $\lambda = 1/6$ en $d = 10$ is

$$E(N(10)) = 5/3 \text{ signalen}$$

Bij de tweede registratiemethode kunnen we de gegevens weergeven in een histogram van de optredende lengten van tussenpozen; we noteren hier de waarden

en de verdeling over de waarden voor de stochast T, de tussenpoos tussen twee opeenvolgende signalen. Er geldt voor de gemiddelde tijdsduur tussen een willekeurig gekozen signaal en het daarop volgende signaal

$$E(T) = 1/\lambda$$

De interpretatie van deze verwachtingswaarde is precies als bij het gemiddeld aantal per tijdinterval. Met $\lambda = 1/6$ is

$$E(T) = 6 \text{ sekonden}$$

Bij een snelheid van optreden van gemiddeld 1/6 signaal per sekonde komen signalen gemiddeld om de 6 sekonden. Dit is bijvoorbeeld het geval als ze afwisselend precies om de 2 en om de 10 sekonden komen. De tussenpozen zijn dan 2 en 10 sekonden, met als overall gemiddelde duur van een tussenpoos 6 sekonden.

Elk van de twee registratiemethoden levert een eigen beschrijvingswijze.

De eerste beschrijvingswijze telt gebeurtenissen —levert een geheel getal— en doet uitspraken over bijvoorbeeld het gemiddeld aantal gebeurtenissen in een interval, algemener gezegd: uitspraken over N. De tweede beschrijvingswijze telt tussenpozen —levert een reëel getal— en doet uitspraken over bijvoorbeeld de gemiddelde tussenpoos, algemener gezegd: uitspraken over T. Er zal in het algemeen heel veel over de verdelingen van de stochasten N en T te zeggen zijn; we zullen voor het Poissonproces deze verdelingen afleiden.

Hoewel beide beschrijvingswijzen volledig equivalent zijn —want een verschijnsel verandert niet als het op een andere manier wordt geregistreerd—, geven ze toch elk een eigen gezicht aan het verschijnsel. De Romeinen beeldden Janus, de god van de tijd, af als een kop met twee aangezichten. Van voren deed het beeld zich anders voor dan van achteren; van de ene kant gezien was het een jongeling, van de andere kant een grijsaard. We zouden de twee registratiemethoden hiermee kunnen vergelijken. Overspringen van de ene naar de andere beschrijvingswijze is een verandering van gezichtspunt; dit is steeds mogelijk, maar gaat gepaard met vervreemdingseffekten als bij een Januskop: men wordt wat draaierig.

12.2.5. samenvatting algemeen optreden signalen

De alledaagse uitspraak dat er per tijdseenheid gemiddeld λ signalen komen, blijkt te betekenen:

- er komen in een willekeurig tijdinterval van lengte d (dat is: per tijdinterval van lengte d) gemiddeld $E(N(d)) = \lambda d$ signalen.
- in nader gespecificeerde intervallen van lengte d komen in het algemeen geen λd signalen gemiddeld (aanstonds zien we dat dit bij signalen volgens een Poissonproces wel het geval is).

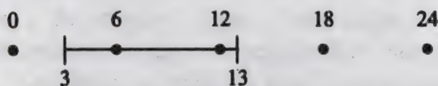
- signalen komen gemiddeld om de $1/\lambda$ tijdseenheden.
- de kans dat er in een willekeurig tijdinterval van lengte d een signaal optreedt is vrijwel λd, mits d klein genoeg.

12.3. POISSONPROCES

12.3.1. basiseigenschap Poissonproces

Nu we weten wat er met 'per interval' wordt bedoeld, wordt er meer duidelijk. Het is —algemeen gesproken— onmogelijk om op grond van alleen het gemiddeld aantal signalen per tijdseenheid aan te geven hoeveel signalen er in heel bepaalde, nader gespecificeerde tijdintervallen gemiddeld optreden.

Als het gemiddeld aantal signalen per tijdinterval van lengte 10 sekonden gelijk is aan 5/3, zal het niet mogelijk zijn af te leiden hoeveel signalen er gemiddeld optreden in intervallen van 10 sekonden, die 3 sekonden na het optreden van een signaal beginnen. Wanneer de signalen met vaste tussenpozen (van 6 sekonden dus) arriveren, komen er in die intervallen altijd (en dus ook gemiddeld) 2 signalen (figuur 12.2).

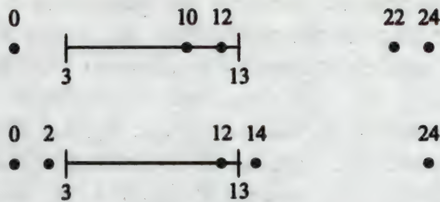


Figuur 12.2. Signalen (•) met vaste tussenpozen van 6 sekonden. Aantal signalen in de 10 sekonden vanaf 3 sekonden na een signaal (tijdstip 0).

Als ze om en om na precies 2 en precies 10 sekonden arriveren, komen er in die intervallen echter even vaak precies 1 als precies 2, dus gemiddeld 3/2 signaal (figuur 12.3). Het gemiddeld aantal signalen in deze intervallen van 10 sekonden is duidelijk niet gelijk aan 5/3, het gemiddeld aantal per tijdinterval van lengte 10. In andere intervallen van lengte 10, bijvoorbeeld die direkt na het optreden van een signaal beginnen, of 7 sekonden later, komt gemiddeld ook een ander aantal signalen. Om het gemiddeld aantal signalen in hele specifieke intervallen te bepalen is het dus nodig (veel) meer over het patroon van aankomsten van signalen te weten dan alleen het gemiddeld aantal per tijdseenheid (we zagen dat ook al bij de intervallen 0-10, 30-40 etc. versus 20-30, 50-60 etc. in figuur 12.1).

Het is dus een heel vergaande veronderstelling dat een bepaald patroon van opvolgende signalen zo verloopt dat het gemiddeld aantal signalen over een interval van een bepaalde duur altijd hetzelfde is, onverschillig† om welk nader

† Uitgezonderd natuurlijk specificaties als 'er trad in het interval al een signaal op'.



Figuur 12.3. Signalen (•) met vaste tussenpozen van om en om 2 en 10 sekonden. Aantal signalen in de 10 sekonden vanaf 3 sekonden na een signaal (tijdstip 0). Beide mogelijkheden komen even vaak voor.

gespecificeerd interval of type interval het gaat. Het gemiddeld aantal in een interval van duur d zou altijd λd zijn, met λ als de snelheid van optreden van signalen. Zo'n patroon moet volledig gespecificeerd zijn door het gemiddeld aantal signalen per tijdseenheid!

Toch blijkt in de praktijk van alledag dat de veronderstelling verrassend vaak erg goed op gaat. Signalen, die volgens zo'n patroon optreden, worden signalen volgens een *Poissonproces* genoemd, of signalen die door een Poissonproces gegenereerd worden. De genoemde veronderstelling is de *basiseigenschap* van het Poissonproces. Deze kenmerkende eigenschap zegt dat over elk interval van een zekere lengte het gemiddeld aantal signalen hetzelfde is. Door deze eigenschap onderscheidt het Poissonproces zich wezenlijk van alle andere processen. Poissonprocessen spelen bij de prestatie-analyse van computersystemen een grote rol.

De basiseigenschap van het Poissonproces lijkt heel bijzonder te zijn, als men er wat langer over nadenkt. Wanneer er in 1 uur gemiddeld 600 signalen optreden, komen er gemiddeld in 10 sekonden 5/3 signalen binnen, zo zagen we. Zoveel treden er bij een Poissonproces ook gemiddeld op in tijdintervallen van 10 sekonden, direkt na de binnenkomst van een signaal. Maar evenveel treden er bij zo'n proces gemiddeld op in tijdintervallen van lengte 10 sekonden, die 1 sekonde na de binnenkomst van een signaal beginnen, of 10 sekonden na de binnenkomst van een signaal, of op het hele uur en het halve uur beginnen, of bij het verspringen van het weerglas, of gerekend vanaf het kraaien van de eerste haan. Speciale intervallen bestaan er bij Poissonprocessen kennelijk volgens de basiseigenschap 'per definitie' niet. Het is niet verwonderlijk dat een Poissonproces wel een *'volmaakt random'* proces wordt genoemd. Het beschrijft een *'zuiver lukraak'* optreden van signalen; het is van zichzelf zo lukraak, dat er nooit iets 'speciaal' is. We zijn benieuwd hoe zo'n proces er uit ziet.

12.3.2. redeneeroefening

We proberen heel algemeen de kans te vinden op één of meer signalen in een tijdinterval van een gegeven eindige lengte (niet noodzakelijk heel kort, zoals in §12.2.3).

Er treden signalen op met een snelheid van gemiddeld λ signalen per tijdseenheid. Wat zal er in een lukraak tijdinterval van, zeg, lengte t gebeuren?

Het tijdinterval kan worden opgesplitst in allemaal even lange intervalletjes van lengte dt (dt heel klein), dat zijn dus t/dt intervalletjes. Voor ieder van die intervalletjes apart is er een kans van λdt op een signaal, de kans op geen signaal is $1-\lambda dt$. De kans op meerdere signalen in het intervalletje dt is nihil.

In het tijdinterval treden soms precies k signalen op. We willen dit nu op de volgende wijze zien. Het fenomeen van k signalen zal optreden als er k maal succes optreedt bij t/dt pogingen op succes met een kans van λdt op succes (we zien een signaal maar als een succes).

Deze kans vinden we via de binomiale verdeling. Nu is het bekend uit de statistiek dat de binomiale verdeling bij een hele kleine kans en een daarmee evenredig groot aantal pogingen overgaat in de zogenaamde Poissonverdeling. Dit zal hier het geval zijn, want naarmate dt kleiner wordt, wordt het aantal pogingen t/dt evenredig groter.

Het aantal malen succes bij de pogingen zal dus Poisson verdeeld zijn en de kans op k successen volgt uit deze verdeling. De stochast $N(t)$ beschrijft het aantal signalen in het lukrake tijdinterval t, dit aantal komt overeen met het aantal successen. Deze stochast is dus Poisson verdeeld, de kansverdeling voor $N(t)$ is volgens deze verdeling

$$Pr(N(t)=k) = \frac{E(N(t))^k}{k!} e^{-E(N(t))}$$

met $E(N(t))=\lambda t$, zoals is ingebracht en overeenkomt met (aantal $= t/dt$) $\times$ (kans op succes $= \lambda dt$). Dus

$$Pr(N(t)=k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

We hebben hier de volledige kansverdeling van de stochast $N(t)$ gevonden en wel uitgedrukt in het produkt van de snelheid van optreden λ, maal als enige parameter de lengte van het tijdinterval. De kans op het optreden van een bepaald aantal signalen binnen een willekeurig interval van lengte t zou bij een snelheid van optreden van λ signalen per t.e. verdeeld zijn volgens een Poissonverdeling met parameter λt!

12.3.3. Poissonverdeling en Poissonproces

Met de getoonde afleiding hebben we uitgaande van de *verwachtingswaarde* $E(N(t))=\lambda t$ de *kansverdeling* $Pr(N(t)=k)$ gevonden. Een kansverdeling bevat veel meer informatie dan de verwachtingswaarde alleen, dat weten we maar al te goed. De extra informatie moet bij de afleiding zijn binnengeslopen. Waar gebeurt dit?

We zagen dat het correct is te stellen dat de kans op een signaal in het lukrake tijdinterval dt gelijk is aan λdt, pas bij strikte synchronisatie van signalen zal dit onjuist zijn. Daar gingen we dus niet buiten ons boekje.

Het kernpunt in de afleiding —en voor het algemene geval de fout— zit echter in de veronderstelling dat voor elk van de opeenvolgende intervalletjes uit het lukrake interval de kans op een signaal λdt is. Wat er in een intervalletje kan gebeuren zou onafhankelijk zijn van wat zich in een ander intervalletje afspeelt. Op dit punt is de gegeven afleiding niet correct. De intervalletjes zijn geen lukrake intervalletjes omdat ze opvolgend zijn!

De herleiding is echter volledig juist bij signalen volgens een Poissonproces. De basiseigenschap van zo'n proces zegt dat in elk interval van lengte dt, waar het interval ook ligt, er steeds gemiddeld λdt signalen optreden, of (omdat dt klein genoeg is) dat de kans op een signaal steeds λdt is (zie §12.2.3). Het is de basiseigenschap van het Poissonproces, die maakt dat de in de afleiding gehanteerde veronderstelling bij een Poissonproces wel correct is. Voor elk opeenvolgend intervalletje is dan de kans op een signaal echt λdt.

Er geldt dus niet voor elk patroon van signalen dat de stochast $N(t)$ Poisson verdeeld is. Het lukt alleen bij signalen volgens een Poissonproces om uit de gemiddelde waarde voor $N(t)$ de hele kansverdeling voor $N(t)$ af te leiden! Voor signalen volgens een Poissonproces laat de 'redeneeroefening' zien dat $N(t)$ Poissonverdeeld is volgens

$$Pr(N(t)=k) = \frac{(\lambda t)^k}{k!}e^{-\lambda t}$$

We kunnen dan zelfs nog verder gaan. Er is bij een Poissonproces geen verschil in verdelingsfunktie tussen $N(t)$: het aantal in een willekeurig interval van lengte t, en $M(t)$: het aantal in een heel bepaald interval van die lengte. Dit weer op grond van de basiseigenschap van het Poissonproces: voor elk interval gaat de aangegeven redenering op en die voert telkens tot de Poissonverdeling. Voor het Poissonproces is dus niet alleen het gemiddelde aantal, maar ook de verdeling van de aantallen signalen voor elk interval dezelfde.

We zullen bij het Poissonproces vasthouden aan de notatie $N(t)$, ook als er eigenlijk $M(t)$ bedoeld is; er is bij het Poissonproces immers toch geen verschil (analoog voor T).

We hebben over de verdeling van het aantal signalen voor het Poissonproces dus het volgende gevonden. De stochast $N(t)$, het aantal signalen in een interval met lengte t —welk interval van lengte t dan ook—, is Poissonverdeeld. De verdeling hangt alleen af van de lengte van het interval en de snelheid van optreden. Voor $N(t)$ geldt:

$$E(N(t)) = \lambda t$$

$$Pr(N(t)=k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$$

Merk op dat de Poissonverdeling veel te maken heeft met het Poissonproces, maar verwar Poissonverdeling en Poissonproces niet met elkaar.

De kans op geen signaal ($k = 0$) is dus $e^{-\lambda t}$, op één of meer signalen $1 - e^{-\lambda t}$. Als in een Poissonproces gemiddeld 1/6 signaal per sekonde optreedt, is de kans op drie signalen in een tijdinterval van lengte tien sekonden gelijk aan

$$Pr(N(10)=3) = \frac{(5/3)^3}{3!} e^{-5/3}$$

en de kans op geen signaal is

$$Pr(N(10)=0) = e^{-5/3}$$

12.3.4. vorm Poissonverdeling

De variantie van de Poissonverdeling volgt zonder meer uit de kansverdeling. Er is uit de statistiek bekend dat deze variantie gelijk is aan de verwachtingswaarde. Bij een Poissonproces geldt naast $E(N)=\lambda t$ ook

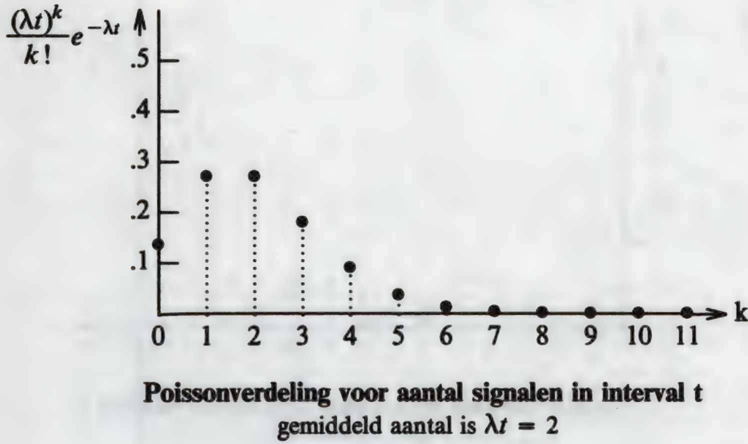
$$VAR(N(t)) = \lambda t$$

en de standaarddeviatie is

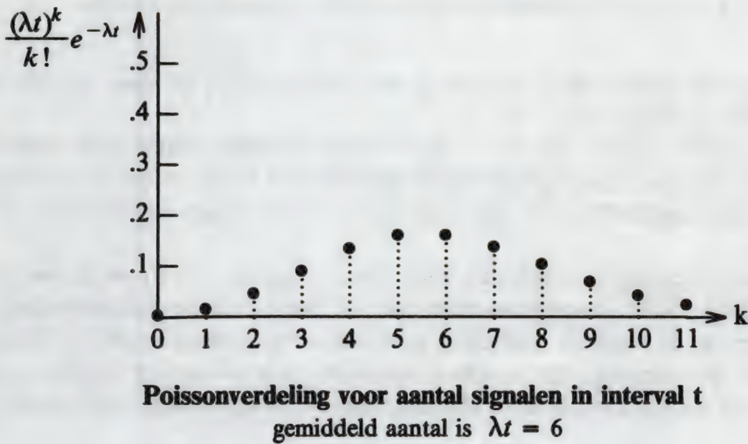
$$\sigma(N(t)) = \sqrt{\lambda t}$$

De Poissonverdeling van het aantal signalen in een tijd t is een discrete verdeling over de niet-negatieve gehele getallen; het aantal signalen dat in een bepaalde tijd optreedt is altijd een geheel getal. We bekijken hier de vorm van deze belangrijke verdeling (meer informatie is in de leerboeken statistiek te vinden).

In de figuren 12.4 en 12.5 is de verdeling weergegeven voor diverse gemiddelde aantallen λt. Bij een kleine waarde voor de verwachtingswaarde λt, bijvoorbeeld een waarde $0 \leqslant \lambda t \leqslant 1$, kan de Poissonverdeling onmogelijk erg symmetrisch zijn: er is maar één enkele mogelijkheid links van de gemiddelde waarde λt, namelijk 0 signalen, alle andere mogelijkheden liggen rechts van het gemiddelde. De verdeling blijkt voor kleinere waarden van λt bij numerieke uitwerking inderdaad flink scheefgetrokken te zijn naar kleine waarden van het aantal N.



Poissonverdeling voor aantal signalen in interval t
gemiddeld aantal is $\lambda t = 2$



Poissonverdeling voor aantal signalen in interval t
gemiddeld aantal is $\lambda t = 6$

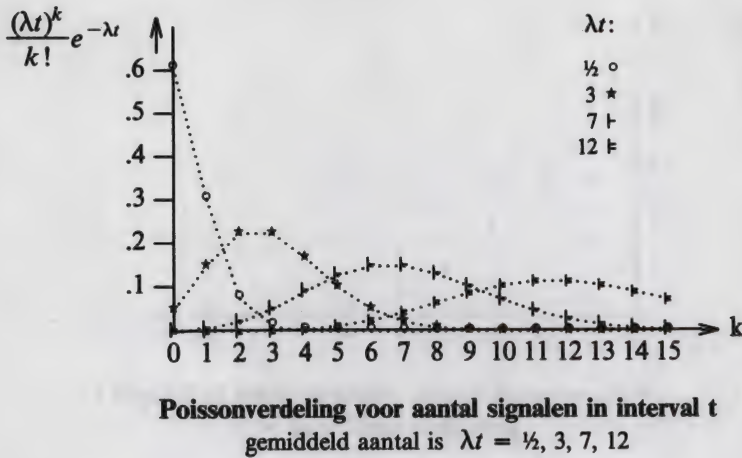
Figuur 12.4. Vorm Poissonverdeling. Sterk asymmetrisch voor klein gemiddeld aantal, vrijwel symmetrisch voor gemiddeld aantal veel groter dan 1.

Voor (bijvoorbeeld) $\lambda = 1/6$ en $t = 6$ met als gemiddelde waarde $\lambda t = 1$, is

$$E(N(6)) = \lambda t = 1$$

$$Pr(N(6)=0) = Pr(N(6)=1) = e^{-1} = 0.37$$

$$Pr(N(6)=i) = e^{-1}/i! = 0.37/i! \quad (i \geqslant 1)$$



Figuur 12.5. Vorm Poissonverdeling voor diverse gemiddelde aantallen.

De kans op een waarde links van het gemiddelde is 0.37, de kans op een waarde rechts van het gemiddelde is 0.26.

Voor grotere waarden van λt is de Poissonverdeling echter wel vrijwel symmetrisch, er zijn dan kennelijk genoeg mogelijkheden links, tussen 0 en het gemiddelde λt, om vergelijkbaar te zijn met de oneindig lange staart rechts van het gemiddelde.

De Poissonverdeling verschilt dus voor kleine waarden van λt sterk van een normale verdeling. Voor grotere waarden ($\lambda t > 5$) heeft ook de Poissonverdeling een klokvorm en is de normale verdeling met verwachtingswaarde λt en variantie λt een redelijke benadering. De statistiek verschaft gedetailleerde middelen om na te gaan wanneer de benadering door een normale verdeling zinvol (dus 'terecht') is.

12.3.5. negatief exponentiele verdeling voor tussenpoos

Het Poissonproces is, als elk optreden van signalen, ook te beschrijven via de tussenpozen van twee opeenvolgende signalen. Dit andere gezicht van het Poissonproces is ook in formules te vangen. De verdeling over de stochast T, de tussenpoos, is voor een Poissonproces rechtstreeks af te leiden uit de snelheid van optreden van signalen.

We merken eerst op (a) dat een interval van lengte t, dat start na het optreden van een signaal, voor een Poissonproces geen bijzonder interval is, het gemiddelde aantal signalen in dit interval is λt en de kans dat er geen enkel signaal in dit interval is, luidt $Pr(N(t)=0) = e^{-\lambda t}$.

Hierna bedenken we dat (b) het optreden van een tussenpoos die groter is dan t, volmaakt overeenkomt met het optreden van (c) geen enkel signaal in het aangegeven tijdinterval; de kansen op het optreden van deze 'gebeurtenissen' kunnen dus niet verschillen.

De kans op (c) is, volgens overweging (a), $e^{-\lambda t}$; dus de kans op (b) moet ook $e^{-\lambda t}$ zijn. Of netjes geformuleerd

$$Pr(T>t) = e^{-\lambda t}$$

Hier staat de cumulatieve kansverdeling $Pr(T \leqslant t) = 1 - e^{-\lambda t}$ en dus ook de kansverdeling voor de stochast T, de tussenpoos.

Door de basiseigenschap van de Poissonverdeling te benutten hebben we, uitgaande van de verdeling van $N(t)$ —de ene kant van de Januskop— de kansverdeling van T —de andere kant van de Januskop— verkregen!

Deze continue kansverdeling voor de tussenpoos is in de statistiek welbekend, het is de negatief exponentiële verdeling (figuur 12.6).

We hebben dus gevonden dat de stochast T, de tussenpoos tussen twee opeenvolgende signalen, bij signalen volgens een Poissonproces negatief exponentieel verdeeld is. De verdeling hangt alleen van de snelheid λ van optreden van signalen af. Voor de stochast T geldt:

$$Pr(T>t) = e^{-\lambda t}$$
$$E(T) = 1/\lambda$$

Als in een Poissonproces gemiddeld 1/6 signaal per sekonde optreedt, is de kans op een tussenpoos tussen twee opeenvolgende signalen, die langer is dan 9 sekonden, gelijk aan

$$Pr(T>9) = e^{-9/6} = \frac{1}{e\sqrt{e}} = 0.22$$

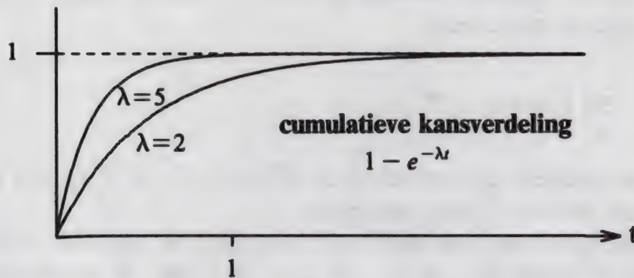
12.3.6. vorm negatief exponentiele verdeling

De negatief exponentiële verdeling wordt rechtstreeks gerepresenteerd door de cumulatieve kansverdeling in de vorm $Pr(T>t) = e^{-\lambda t}$. Met de kansdichtheidsfunktie $f(x)$, gedefinieerd via

$$Pr(t \leqslant T \leqslant t+dt) = f(t)dt$$

wordt dit geschreven als

$$\int_t^\infty dx\, f(x) = e^{-\lambda t}$$



Figuur 12.6. Poissonproces: kans op een tussenpoos kleiner dan t als funktie van t.

De integraal loopt over de mogelijke waarden voor T, dus van t tot oneindig. Differentiëren naar t van linker- en rechterkant (naar de ondergrens van de integraal) levert

$$f(t) = \lambda e^{-\lambda t}$$

De continue kansdichtheidsfunktie van de negatief exponentiële verdeling is dus een op de positieve t-as overal exponentieel dalende funktie, dalend vanaf de waarde λ voor $t = 0$ tot de waarde 0 voor $t = \infty$ (zie figuur 12.7).

De verwachtingswaarde van de negatief exponentiële verdeling is volgens de definitie van kansdichtheid (alleen positieve waarden voor T zijn mogelijk, dus een integraal van 0 tot oneindig):

$$E(T) = \int_0^\infty dt\, t\, \lambda\, e^{-\lambda t}$$

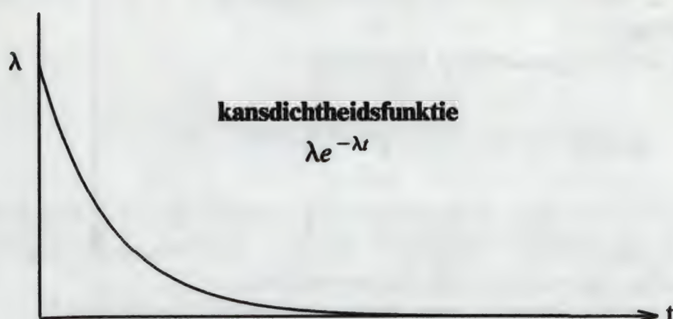
en na integratie

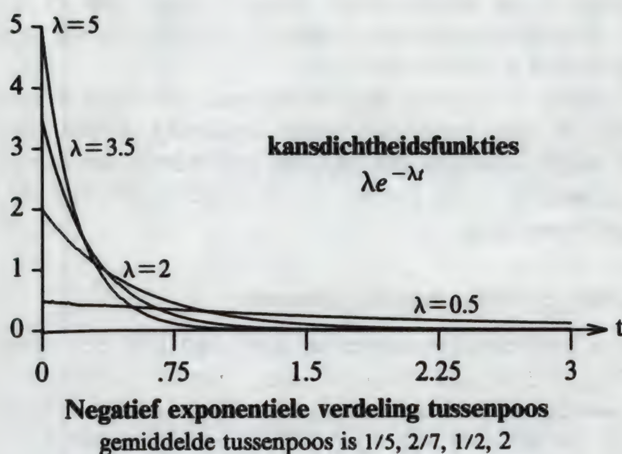
$$E(T) = 1/\lambda$$

Voor de variantie geldt:

$$VAR(T) = \int_0^\infty dt(t-1/\lambda)^2 \lambda e^{-\lambda t} = 1/\lambda^2$$

De uitkomst voor de verwachtingswaarde van de tussenpoos is niet verbazend. Als





Figuur 12.7. Poissonproces: kansverdeling tussenpozen signalen.

er gemiddeld 10 signalen per minuut komen is $\lambda = 1/6$ signalen per sekonde. De gemiddelde tussenpoos $E(T)$ is dan volgens bovenstaande uitkomst $1/\lambda = 6$ sekonden. Ook de gewone ervaring leert dit, we memoreerden het zojuist in §12.2.4.

Uit de waarde van de variantie blijkt dat inderdaad de variatiecoëfficiënt van de negatief exponentiële verdeling 1 is, dit gebruikten we eerder in het hoofdstuk RESTDUREN EN WACHTDUREN (§8.2.1).

De kans op een tussenpoos korter dan de gemiddelde tussenpoos is

$$Pr(T<1/\lambda) = 1-e^{-1} = 0.63$$

Dus 63% van de tussenpozen is korter dan gemiddeld —en niet 50%: de verdeling is niet symmetrisch.

Korter dan de halve gemiddelde tussenpoos is

$$Pr(T<1/(2\lambda)) = 1-e^{-1/2} = 0.39$$

Bijna twee van de vijf negatief exponentieel verdeelde tussenpozen zijn korter dan de helft van de gemiddelde tussenpoos: een groot deel van de tussenpozen is relatief kort vergeleken met de gemiddelde tussenpoos.

De kans op een tussenpoos langer dan tweemaal het gemiddelde is

$$Pr(T>2/\lambda) = e^{-2} = 0.14$$

Slechts 14% van de tussenpozen haalt het dubbele van de gemiddelde waarde; deze tussenpozen vallen in de exponentieel dalende staart van de kansdichtheidsverdeling $\lambda e^{-\lambda t}$. Eenzelfde berekening levert op dat 90% van alle tussenpozen kleiner is dan $2.3/\lambda$ en 95% is kleiner dan $3/\lambda$.

In gewone woorden samengevat: bij tussenpozen, die negatief exponentieel verdeeld zijn, treffen we veel korte aan, naast aanzienlijk minder van 'middelbare lengte' (van een lengte van ongeveer $1/\lambda$) en heel weinig van grotere lengte. De kans op een betrekkelijk korte tussenpoos is altijd groter dan de kans op een betrekkelijk lange tussenpoos.

12.3.7. samenvatting eigenschappen Poissonproces

We weten nu hoe het Poissonproces er uit ziet. Het kan zich op twee manieren presenteren. Of het doet zich voor als iets, waarbij gedurende een bepaalde periode geteld wordt, de relevante stochast is N, verdeeld volgens de discrete Poissonverdeling. Of het doet zich voor als iets, waarbij tijdsduren worden gemeten, de relevante stochast is T, verdeeld volgens de continue negatief exponentiële verdeling. De Poissonverdeling en de negatief exponentiële verdeling horen bij elkaar, het zijn de twee kanten van de Januskop van het Poissonproces. Vinden we ergens onderling onafhankelijke tussenpozen die negatief exponentieel verdeeld zijn, dan komen de bijbehorende 'signalen' van een Poissonproces. Tellen we gedurende enige tijd aantallen, die Poisson verdeeld zijn, dan tellen we signalen volgens een Poissonproces.

Als er per tijdseenheid gemiddeld λ signalen komen, is λt het gemiddelde aantal signalen in elk lukraak tijdinterval van lengte t. De gemiddelde tussenpoos tussen signalen is $1/\lambda$. Voor een Poissonproces zijn niet alleen deze gemiddelden, maar zelfs de verdelingen aan te geven voor elk —niet per se lukraak— interval. Bij een Poissonproces komen signalen op lukrake momenten, voor een Poissonproces heeft geen enkel tijdinterval iets speciaals.

De kans op k signalen in t tijdseenheden is

$$Pr(N(t)=k) = \frac{(\lambda t)^k}{k!}e^{-\lambda t}$$

De kans op een tussenpoos langer dan t is

$$Pr(T>t) = e^{-\lambda t}$$

De variantie in $N(t)$ is λt, de variantie in T is $1/\lambda^2$.

In de meeste studieboeken wordt het Poissonproces veel strikter ingevoerd dan hier is gebeurd. Het wordt daar gebracht als een proces, waarvoor eerst een lijst van mathematisch geformuleerde eigenschappen wordt gepostuleerd, op grond waarvan men dan tot de aangegeven verdelingen van N en T besluit. Men neemt de tijd als parameter —want de tijd maakt onderscheid— en gebruikt de begrippen onafhankelijk en stationair (en stopping time) om aan te geven dat 'elk' interval zich gelijk gedraagt. De door ons aangegeven weg via de 'basiseigenschap' laat heel duidelijk het alledaagse van het Poissonproces zien, ook deze route kan zo gepreciseerd worden dat er wiskundig geen vinger tussen te krijgen is —zie bijvoorbeeld het klassieke boek van ÇINLAR. Een eenvoudige precieze definitie van het Poissonproces staat in §13.4.

12.4. BELANG POISSONPROCES

Het Poissonproces, met zijn eigenzinnige lukrake karakter, speelt een hoofdrol bij het beschrijven van complexe verschijnselen, zolang daarin niet alle details worden meegenomen. In het volgende hoofdstuk gaan we nog wat nader in op de eigenschappen van dit proces, maar eerst is het tijd om naar de praktijk te gaan kijken.

Allereerst dit. Signalen volgens een Poissonproces komen op lukrake momenten binnen. Als door die signalen iets wordt waargenomen, funktioneren ze als de lukrake waarnemer! Wanneer opdrachten ergens binnen komen volgens een Poissonproces, treffen ze gemiddeld de situatie aan, die een lukrake waarnemer ook gemiddeld zal vinden. Dit is belangrijk, want over wat de lukrake waarnemer gemiddeld ziet hebben we in het voorgaande al heel wat informatie verzameld; de lukrake waarnemer treft de '(tijd)gemiddelde' toestand aan. Dit aspekt van het Poissonproces is van groot belang bij de wachtrijmodellen voor computersystemen (vergelijk §15.3). Deze eigenschap wordt in de literatuur wel het PASTA-principe genoemd: Poisson Arrivals See Time Averages.

Het Poissonproces blijkt voor de verdeling van tussenpozen in de tijd dezelfde rol te spelen als de bekende normale verdeling voor de verdeling van tussenafstanden in de (eendimensionale) ruimte. Signalen zullen vaak door heel verschillende

oorzaken ontstaan en de tussenpozen tussen signalen met dezelfde oorzaak kunnen vreemde verdelingen hebben. Deze verdelingen bepalen samen de 'overall' verdeling van de tussenpozen van signalen. Als er (voldoende) veel onafhankelijke oorzaken zijn met een eigen verdeling van de tussenpozen, is deze 'overall' verdeling van de tussenpozen tussen signalen negatief exponentieel, net zoals de som van een (voldoend) groot aantal afstanden normaal verdeeld is, hoe de verdelingen van de afzonderlijke afstanden ook zijn. Deze belangrijke eigenschap kan voor een samenstel van heel veel renewal-processen (§13.4) precies worden bewezen.

In computersystemen worden veel hulpbronnen (shared resources) —CPU, geheugen, randapparaten— 'eerlijk' verdeeld onder heel verschillende mededingers. Er kan om uiteenlopende redenen bij de hulpbron iets 'gebeuren'. De aangehaalde stelling bevestigt dat we het optreden van zo'n voorval als 'lukraak' mogen betitelen, zolang we niet in detail treden.

De negatief exponentiële verdeling kan ook optreden wanneer er niet direkt een Poissonproces aan gekoppeld is. Zo zijn bijvoorbeeld de serviceduren van bepaalde typen klanten negatief exponentieel verdeeld. Of de grootten van geheugensegmenten.

In het hoofdstuk RESTDUREN EN WACHTDUREN bekeken we hoelang het na het optreden van een lukrake waarnemer duurt tot de huidige toestand voorbij is; die tijd is de 'restduur'. De lukrake waarnemer kan corresponderen met een signaal uit een Poissonproces. We vonden in §8.2 de eenvoudige uitdrukking voor de gemiddelde restduur. Als de toestandsduren negatief exponentieel verdeeld zijn is de gemiddelde restduur gelijk aan de gemiddelde toestandsduur, omdat C dan 1 is (§12.3.6). Bij toestandsduren die tussenpozen zijn uit een Poissonproces is de restduur negatief exponentieel verdeeld. Op grond van de basiseigenschap van het Poissonproces is dat zelfs het geval als de resterende duur niet vanaf een lukraak moment wordt gerekend (vergelijk §12.3.5). De basiseigenschap is weer machtig en zegt ook dat het voor een Poissonproces vanaf elk moment gemiddeld nog de gemiddelde tussenpoos duurt totdat het volgende signaal komt. In het volgende hoofdstuk gaan we dit aan de hand van de verdelingsfunkties expliciet na (§13.2).

In de voorgaande hoofdstukken zagen we telkens dat een beschrijving veel eenvoudiger wordt als bekend is dat iets 'lukraak' gebeurt. Ook in dit hoofdstuk zien we dat 'lukraak' als een zuurdesem allerlei redeneringen doortrekt. Het Poissonproces, dat volmaakt lukraak optreedt, moet dus wel van groot belang zijn voor de prestatie-analyse van computersystemen.

De Poissonverdeling treedt inderdaad veelvuldig op bij wat er zich in en om de computer in de tijd afspeelt. Bij vele devices zijn de aankomstpatronen van accessen Poissonverdeeld. Soms zijn ook de serviceduren bij de devices negatief exponentieel verdeeld. Of de tussenpozen bij het verlaten van het device. De denktijden van terminalisten zijn nogal eens keurig verdeeld volgens de negatief exponentiële verdeling (de tijd die een slokje koffie kost buiten beschouwing gelaten). In de analyse van de prestaties van protocollen voor computernetwerken wordt vaak uitgegaan van negatief exponentieel verdeelde berichtlengten. Kortom,

in het onderzoek van computersystemen zullen Poissonprocessen een belangrijke rol spelen. Er volgt hier een aantal simpele inleidende toepassingen.

12.5. VOORBEELDEN

12.5.1. voorbeeld: aanbod

Een bepaald type opdracht wordt een rekenmachine aangeboden met tussenpozen die negatief exponentieel verdeeld zijn. Gemiddeld wordt één zo'n opdracht aangeboden per 24 minuten 'continu in bedrijf zijn' van de machine.

Het gemiddeld aantal opdrachten van dit type dat in 72 minuten continubedrijf wordt aangeboden is 3. De snelheid van aankomst van het signaal 'er is een opdracht' is $1/24$ per minuut, de duur van het tijdinterval is 72 minuten; het produkt is inderdaad 3.

De tussenpozen tussen het aanbieden van deze opdrachten zijn gemiddeld 24 minuten met een standaarddeviatie van 24 minuten (§12.3.6).

De kans dat het tussen de 48 en 72 minuten duurt voordat de volgende opdracht van dit type wordt aangeboden is $e^{-2}-e^{-3} = 0.086$.

12.5.2. voorbeeld: denktijden

De denktijden aan een bepaalde terminal zijn negatief exponentieel verdeeld met een gemiddelde denktijd van 20 sekonden. Van deze denktijden is dus 14% langer dan 40 sekonden en 39% is korter dan 10 sekonden (§12.3.6). Zelfs (of slechts) 10% is langer dan $2.3\times20 = 46$ sekonden en 5% is langer dan 60 sekonden. Door oorzaken als kletsen en dagdromen zullen deze cijfers in de praktijk iets hoger liggen.

Als een terminalist 20 sekonden gedacht heeft, kost het hem gemiddeld nog eens 20 sekonden voordat hij zijn opdracht inbrengt (de resterende duur van de toestand 'denkend' is gemiddeld de gemiddelde denktijd; §12.4, §13.2).

12.5.3. voorbeeld: volraken buffer

Een randapparaat van een computer verwerkt opdrachten, deze komen binnen volgens een Poissonproces. Er komen in een sekonde gemiddeld 3 opdrachten binnen. Binnenkomende opdrachten worden gebufferd voordat ze door het randapparaat gelezen en verwerkt worden. De buffer kan maximaal 3 opdrachten herbergen.

Op een moment waarop er geen enkel bericht in de buffer is valt het randapparaat uit. Na enige tijd zit de buffer vol en treedt er 'overflow' op, doordat een opdracht niet geplaatst kan worden. Een sekonde na het uitvallen is de kans op een volle buffer:

$$Pr(N(1)\geqslant 3) = 1-(1+3+9/2)e^{-3} = 0.58$$

Na het volraken van de buffer zal de eerstkomende opdracht overflow veroorzaken. De kans dat deze ramp na het vollopen van de buffer nog langer dan een halve sekonde uitblijft, is (§12.4, §13.2)

$$Pr(T > 1/2) = e^{-3/2} = 0.22$$

12.5.4. voorbeeld: levensduur

De 'levensduur' van een onderdeel van een machine is de tijd tussen het inzetten van het onderdeel en het vervangen van het onderdeel omdat het storingen vertoont. Als de levensduur negatief exponentieel verdeeld is bestaat er een vaste kans, onafhankelijk van de voorgeschiedenis, op het optreden van het signaal 'kapot'. Dit is een goede benadering in situaties waarin het effekt van slijtage, van ouderdomsverschijnselen, gering is in vergelijking met lukraak optredende externe oorzaken zoals stroomfluktuaties, opeenhopingen van stof, enzovoort. Bij veel onderdelen is het effekt van veroudering alleen van belang direkt na het inzetten (zwak onderdeel: fabrieksfout) en als het al erg lang meegaat.

Een bepaalde machine wordt dag en nacht gebruikt. Een onderdeel van de machine funktioneert gemiddeld een week zonder storingen te vertonen. Zodra zich een storing bij het onderdeel voordoet wordt het in zijn geheel vervangen. Laten we aannemen dat het optreden van storingen een zuiver lukraak proces is, dus dat de levensduur van het onderdeel negatief exponentieel verdeeld is.

De kans dat een nieuw onderdeel het minstens een week uithoudt is $1-0.63 = 0.37$, want voor de negatief exponentiële verdeling is er een kans van 0.63 op een waarde kleiner dan de gemiddelde waarde (§12.3.6).

Als de vervanger het drie weken heeft uitgehouden duurt het nog gemiddeld een week voordat ook hij uitvalt.

12.5.5. voorbeeld: interrupt-monitoring (vervolg opgave 7.1)

De centrale verwerkingseenheid (CPU) kan interrupts ontvangen, die afkomstig zijn van randapparaten. De CPU verwerkt deze interrupts in een interrupt-afwikkelingsroutine (-handler). Er zijn drie soorten interrupts. Van alle gegenereerde interrupts is 1/6 deel van soort *I*, de verwerkingsduur van deze interrupts (duur van de routine) is precies 30 msek. Verder is 1/3 deel van soort *II* met een duur van precies 15 msek. en de helft van soort *III* met een duur van precies 20 msek.

Er is geconstateerd dat de CPU bezig is met het afwikkelen van interrupts. Op zo'n moment komt er nòg een interrupt binnen, deze blijft 'hangen'. Interrupts komen binnen volgens een Poissonproces, er komen gemiddeld 2 interrupts in 60 msek: $\lambda = 1/30$ per msek. Een interrupt komt dus op een lukraak moment.

Volgens de basisrelatie voor de kansen op optreden/aantreffen is de kans, dat de

CPU in de interrupthandler van soort I bezig is, niet 1/6, maar

$$p_I = \frac{30/6}{30/6+15/3+20/2} = 1/4$$

Idem $p_{II} = 1/4$ en $p_{III} = 1/2$.

Het duurt gemiddeld nog $15/4+7.5/4+10/2 = 85/8$ msek. voordat de handler is doorlopen en de hangende interrupt zou kunnen worden verwerkt (§8.2).

Er komen gemiddeld $1/30\times20=2/3$ interrupts binnen in de verwerkingsduur van een interrupt van type *III*.

De kans dat meer dan één interrupt binnenkomt is

$$Pr(N(20)>1) = 1-(1+2/3)e^{-2/3} = 0.14$$

12.5.6. vervolg voorbeeld: gebundelde Remote-write's (7.3.5; 8.5.1)

Requests komen volgens een Poissonproces binnen, er komt gemiddeld één request per 40 msek. De Remote-write-1, -2, -3 en -4 operaties duren precies 30, 40, 50 en 60 msek. Deze gegevens, hoe weinig ook maar, maken het mogelijk een behoorlijk beeld te geven van het verwerkingspatroon van Remote-write requests.

In de 50 msek. waarin een Remote-write-3 operatie verwerkt wordt, komen een aantal requests binnen; gemiddeld 5/4 requests ($\lambda = 1/40$, $t = 50$, $\lambda t = 5/4$). De kansen op één, twee, drie en vier requests in deze verwerkingsduur zijn

$$Pr(N(50)=1) = (5/4) \quad e^{-5/4} = 0.36$$

$$Pr(N(50)=2) = \frac{(5/4)^2}{2} e^{-5/4} = 0.22$$

$$Pr(N(50)=3) = \frac{(5/4)^3}{6} e^{-5/4} = 0.09$$

$$Pr(N(50)=4) = \frac{(5/4)^4}{24} e^{-5/4} = 0.03$$

(de kans is het grootst voor één request, dat is in de buurt van het gemiddelde aantal van 5/4 requests).

In deze gevallen wordt na afloop van de Remote-write-3 operatie een Remote-write-1, respectievelijk -2, -3 en -4 operatie uitgevoerd. De kans dat een Remote-write-5 operatie volgt, is zo klein (0.007) dat we deze verwaarlozen, net als de kansen op Remote-write-6, Remote-write-7, enzovoort (we zijn wat lui, eigenlijk is de kans op Remote-write-6 pas echt klein).

Er is ook een kans op geen enkele binnenkomst

$$Pr(N(50)=0) = e^{-5/4} = 0.29$$

Als dit gebeurt raakt de server een poosje idle, want de drie requests die uitstonden zijn gebundeld verwerkt in de ene Remote-write-3 operatie. Na gemiddeld 40 msek. komt er weer een request, die wordt omgezet in een Remote-write-1 operatie. Ook in dit geval zal er dus na de Remote-write-3 operatie als volgende operatie een Remote-write-1 operatie komen. De kans dat er op een Remote-write-3 operatie een Remote-write-1 operatie volgt is dus $Pr(N(50)=0) + Pr(N(50)=1) = 0.36+0.29 = 0.64$.

Het patroon van opvolging van Remote-write operaties kan beschreven worden met een overgangsmatrix. Zojuist hebben we de elementen in de rij die hoort bij Remote-write-3 aangegeven. De kansen op overgangen vanuit Remote-write-1, Remote-write-2 en Remote-write-4 worden net zo bepaald. De overgangsmatrix staat in figuur 12.8.

$$\mathbf{P} = \begin{bmatrix} 0.83 & 0.13 & 0.03 & 0.01 \\ 0.74 & 0.18 & 0.06 & 0.02 \\ 0.64 & 0.22 & 0.09 & 0.03 \\ 0.56 & 0.25 & 0.13 & 0.05 \end{bmatrix}$$

Figuur 12.8. Overgangsmatrix gebundelde Remote-write's.

De relatieve frequenties waarmee Remote-write-i operaties optreden, vinden we als kansen op optreden uit π met $\pi=\pi\mathbf{P}$. Oplossen levert $\pi = (0.80, 0.15, 0.04, 0.01)$ (opgave 12.3).

De kansen op optreden, bepaald uit $\pi=\pi\mathbf{P}$, geven precies aan welk deel van de operaties een Remote-write-i is. Met de overgangsmatrix hebben we het hulpmiddel in handen om uit informatie over het volgordepatroon van Remote-write's te bepalen hoe vaak een bepaalde Remote-write relatief voorkomt!

Het grote belang van '$\pi=\pi\mathbf{P}$' blijkt hier weer. Uit het patroon van 'stuivertje wisselen' wordt informatie gehaald over het relatieve optreden, over de samenstelling van de werklast.

Hier lezen we aan de oplossing voor π af dat de Remote-write-3 operaties maar een heel klein deel uitmaken van de te verwerken werklast, slecht 4% van de uit te voeren operaties. Misschien hadden we verwacht dat het er meer zouden zijn, toen bleek dat er tijdens een Remote-write-3 operatie gemiddeld 5/4 requests binnenkomen.

De verdeling van de verwerkingsduren van de operaties is nu volledig bekend. Er zijn duren van 30, 40, 50 en 60 msek., met kansen van respectievelijk 0.80, 0.15,

0.04 en 0.01. We brachten bij de kennismaking met de gebundelde Remote-write's in §7.3.5 uit BEURTEN EN KANSEN ad hoc in dat deze kansen 0.83, 0.13, 0.04 en 0.0 (etc.) waren. Bij een volkomen lukraak patroon van aanvragen kunnen we deze kansen dus berekenen.

We zien hoe de kennis uit de vorige hoofdstukken RESTDUREN EN WACHTDUREN en ERGODISCHE MARKOVKETENS kan samenvloeien met de informatie die het Poissonproces oplevert. Het 'zuiver lukrake' daarvan maakt veel berekeningen mogelijk. Uitgaande van een Poissonproces voor het aanvragen van de Remote faciliteit en gegeven de verwerkingsduren bij het afwikkelen, hebben we berekend hoe vaak de verschillende vormen van verwerking relatief voorkomen.

De berekening van de gemiddelde wachtduren voor requests uit §8.5.1 gaat hier op omdat requests volgens een Poissonproces op lukrake momenten binnenkomen.

12.6. SAMENVATTING

We gingen algemeen na wat het betekent dat er λ signalen per tijdseenheid komen. Het aantal signalen in een lukraak tijdinterval van lengte d is gemiddeld λd. Signalen komen gemiddeld om de $1/\lambda$ t.e. Signalen volgens het volmaakt lukrake Poissonproces komen op volledig lukrake momenten, het aantal in elk —niet per se lukraak— tijdinterval van lengte d is Poissonverdeeld rond het gemiddelde λd. De tussenpozen tussen signalen uit zo'n proces zijn negatief exponentieel verdeeld.

Wanneer er in een systeem op lukrake momenten iets gebeurt, geven de basisbetrekkingen uit de voorgaande hoofdstukken informatie over (tijd)gemiddelden. In dit hoofdstuk blijkt dat lukrake momenten beschreven kunnen worden met een Poissonproces.

Het Poissonproces is van groot belang voor de prestatie-analyse. Verwerking in een computersysteem is een complex samenspel, daarom gebeurt er globaal gezien veel lukraak, dus volgens een Poissonproces. Hierdoor neemt de kennis van wat er gebeurt sterk toe. Op grond van de basiseigenschap van het Poissonproces zijn de verdelingen van aantallen en tussenpozen steeds bekend. Dit maakt het mogelijk berekeningen die uitgaan van de responsetijdrelaties, de relaties tussen de kansen op optreden/aantreffen en de evenwichtsrelaties verder uit te bouwen en toe te passen.

12.7. OPGAVEN

opgave 12.1: aanbod batch

Op een instituut werden in het verleden programma's gedraaid via een verbinding als remote-terminal op een grote computerconfiguratie. We bekijken situaties, die bij zo'n verwerking optraden. Alle ingeleverde programma's worden om het halve uur verzameld en gedraaid. Uit tellingen is bekend dat er gemiddeld 3 programma's in zo'n periode van een halfuur worden aangeboden.

Veronderstel dat programma's worden ingeleverd volgens een Poissonproces.

- Hoe wordt de veronderstelling getoetst dat de aangeboden aantallen Poisson verdeeld zijn? Lijkt de veronderstelling redelijk?
- Hoe groot is de kans dat het aanbod aan programma's in een half uur groter is dan gemiddeld?
- De operator vindt als hij wil draaien af en toe geen materiaal in het bakje, waarin de te draaien programma's worden gedeponeerd. Neem een normale dag waarop met tussenpozen van een half uur 8 maal gedraaid wordt. De allereerste keer is er steeds materiaal aanwezig (van de vorige avond, vroege ochtend). Bepaal de kans dat de operator minstens één keer geen materiaal aantreft.
- Zojuist is er een programma in het bakje met te draaien programma's gelegd. Bepaal hoe groot de kans is dat het langer dan een kwartier duurt voordat het volgende programma wordt aangeboden.

opgave 12.2: vervolg interrupt-monitoring (12.5.5)

- Hoeveel interrupts komen er gemiddeld bij het afwikkelen van een interrupt in de handler binnen (0.480)?

De stationaire kansen op 3 respectievelijk 2, 1, 0 opdrachten in de buffer zijn volgens berekeningen (vergelijk §16.2) of metingen 0.10, 0.20, 0.30 en 0.40.

- Wat is de kans op een volle buffer 1 sekonde na uitvallen? Hoe groot is de kans dat het daarna minder dan 1 sekonde duurt voordat overflow optreedt?

opgave 12.3: vervolg gebundelde Remote-write's (12.5.6)

De overgangsmatrix uit figuur 12.8 is niet helemaal stochastisch, omdat de mogelijkheid dat er een Remote-write-5 (Remote-write-6, enzovoort) loopt is verwaarloosd.

- Maak de berekening preciezer door Remote-write-5 (Remote-write-6, enzovoort) mee te nemen (gebruik een computer).
- Maak de benadering preciezer door aan te nemen dat de Remote-write-i's voor $i>4$ even lang duren als Remote-write-4.

13

Eigenschappen Poissonproces

13.1. INLEIDING

De basiseigenschap van het Poissonproces karakteriseert het als 'zuiver lukraak'. Het Poissonproces wordt ook wel het proces zonder geheugen genoemd. Deze intrigerende benaming brengt bondig tot uitdrukking wat er singulier is aan het Poissonproces. De karakteristieke eigenschappen van het Poissonproces zijn:

- zuiver lukraak: geen enkel tijdinterval is speciaal, alleen de lengte van het tijdinterval telt.
- zonder geheugen: informatie over wat er al is gebeurd, is irrelevant voor de voorspelling van wat er zal gaan gebeuren.

In dit korte hoofdstuk worden deze eigenschappen van het Poissonproces nader toegelicht. De operationele achtergrond wordt aangestipt.

Het renewal-proces is het eenvoudigste model om 'tussenpozen' exact te beschrijven. Het Poissonproces is een renewal-proces met negatief exponentieel verdeelde tussenpozen.

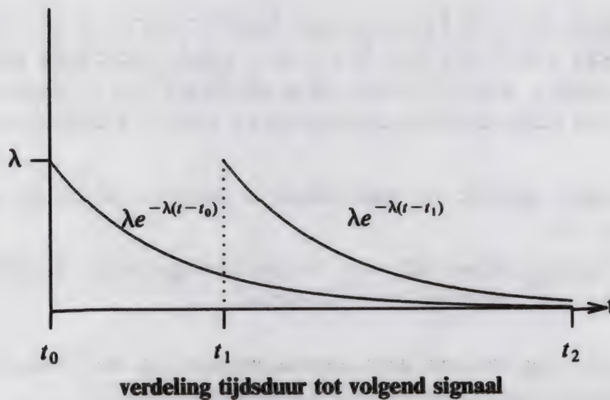
13.2. LUKRAAK ZONDER GEHEUGEN (MATHEMATISCH INTERMEZZO)

Er komen gemiddeld λ signalen per tijdseenheid volgens een Poissonproces. Veronderstel dat op het tijdstip t_0 voor het laatst een signaal optrad. Vanaf dat moment wordt gewacht op het eerstkomende signaal (figuur 13.1). Nu is het al het tijdstip t_1, dus is het $t_1 - t_0$ t.e. geleden dat het laatste signaal kwam. Hoelang moet er verder nog gewacht worden?

Gemiddeld moeten we $1/\lambda$ t.e. wachten als we beginnen met wachten op het moment van optreden van een signaal, immers de gemiddelde tussenpoos tussen de signalen zal $1/\lambda$ zijn. Er is al enige of geruime tijd gewacht $(t_1 - t_0)$. Men zou misschien verwachten dat er nu gemiddeld korter dan $1/\lambda$ t.e. moet worden gewacht op het aanstaande signaal. Dat is bij aankomsten volgens een Poissonproces echter niet correct (vergelijk reeds §8.2). Er zal nog steeds gemiddeld $1/\lambda$ t.e. moeten worden gewacht.

Voor het Poissonproces is ieder tijdinterval gelijkwaardig, zoals de basiseigenschap zegt, of het interval begint op t_0 of op t_1 maakt niets uit. De kans op een signaal in een kort tijdinterval direkt na het laatste optreden, zal gelijk zijn aan de kans op een signaal in een even kort tijdinterval na een moment, waarop het laatste optreden al weer lang tot het verleden behoort. Daardoor zal de gemiddeld te wachten tijdsduur gelijk zijn. Of er vanaf t_0 wordt gewacht of vanaf t_1 maakt geen verschil.

Dit blijkt ook uit de formules voor de kansverdeling (vergelijk figuur 13.1).



verdeling tijdsduur tot volgend signaal

links: vanaf t_0; rechts: vanaf t_1, geen signaal tot aan t_1

Figuur 13.1. Negatief exponentiele verdeling heeft geen geheugen.

De kans dat het langer dan het tijdstip t_2 duurt voordat het signaal komt, is

$$Pr(N(t_2 - t_0) = 0) = e^{-\lambda(t_2 - t_0)}$$

Uitgedrukt in de wachtduur T vanaf t_0 is dit

$$Pr(T > t_2 - t_0) = e^{-\lambda(t_2 - t_0)}$$

Er is al gèwacht vanaf t_0 tot t_1, zonder resultaat. Dus is zeker $T \geqslant t_1 - t_0$. Voor de resterende wachtduur G zoeken we de kansverdeling, we zoeken in concreto met $G = T-(t_1-t_0)$:

$$Pr(G \geqslant t_2 - t_1) = Pr(T-(t_1-t_0) \geqslant t_2 - t_1 \mid T \geqslant t_1 - t_0)$$

Deze voorwaardelijke kans is

$$Pr(G \geqslant t_2 - t_1) = \frac{Pr(T-(t_1-t_0) \geqslant t_2 - t_1 \cap T \geqslant t_1 - t_0)}{Pr(T \geqslant t_1 - t_0)}$$

De teller is, omdat de tweede ongelijkheid uit de eerste volgt,

$$Pr(T-(t_1-t_0) \geqslant t_2 - t_1) = Pr(T \geqslant t_2 - t_0) = e^{-\lambda(t_2-t_0)}$$

terwijl de noemer is

$$Pr(T \geqslant t_1 - t_0) = e^{-\lambda(t_1-t_0)}$$

Delen geeft

$$Pr(G \geqslant t_2 - t_1) = e^{-\lambda(t_2-t_1)}$$

De wachtduur G vanaf t_1 heeft inderdaad dezelfde kansverdeling als T, de wachtduur vanaf t_0. De gemiddelde waarde van G is dus ook $1/\lambda$, er moet nog gemiddeld $1/\lambda$ t.e. gewacht worden. Kennis van de historie —er is al gewacht— verandert niets aan de voorspelling over de toekomst: men moet nog gemiddeld de gemiddelde tussenpoos wachten. Geheugen doet niet ter zake, vandaar de kwalificatie 'geheugenloos' voor het Poissonproces.

Deze afleiding van het geheugenloos opereren van het Poissonproces kunnen we ook als volgt interpreteren. Van de hele negatief exponentieel verdeelde kansverdeling over T is op het tijdstip t_1 alleen de staart rechts van t_1 relevant. Deze staart wordt tot een bona fide kansverdeling gemaakt door te delen door het oppervlak van de staart, dat is de totale kans rechts van t_1, dus $Pr(T \geqslant t_1 - t_0)$. Door deze deling wordt de totale resterende kans keurig 1.

Het opmerkelijke is dat door dit 'opblazen' (delen door getal <1) de oorspronkelijke kansverdeling terugkeert. Iedere staart van een negatief exponentiële verdeling heeft dus overal exact dezelfde vorm, of er nu een lang of een kort eind wordt bekeken; de negatief exponentiële verdeling is overal gelijk.

Zonder geheugen betekent dus, net als bij de Markovketens, dat de kans dat iets gebeurt niet afhangt van wat er tevoren gebeurde (§9.3, §9.4.1).

Het geheugenloos opereren van een Poissonproces maakt dat veel gegevens niet direkt relevant zijn. Daardoor is het dikwijls mogelijk in ingewikkelde situaties eenvoudige voorspellingen te doen. Het is voldoende te weten hoe vaak iets per tijdseenheid gebeurt, mits lukraak.

Hierbij lijkt het soms dat de eenvoud van het Poissonproces uitkomsten oplevert, die tegen de intuïtie ingaan; tenminste zolang die intuïtie niet geschoold is in 'lukraak'. Men is aanvankelijk niet bereid allerlei bijkomende invloeden slechts impliciet op te nemen door 'blij te zijn' dat deze faktoren maken dat de gang van zaken globaal gezien sterk op een 'lukraak verloop' lijkt.

Een zuiver lukraak aspekt van het Poissonproces is ook het volgende. Veronderstel dat in een periode van lengte L precies één signaal is opgetreden. Dit signaal kwam ergens in de tijdsduur L. Er wordt achteraf gevraagd, wanneer dit mag zijn geweest.

Als alles volstrekt lukraak gebeurt, moet de kans dat het in de eerste t t.e. was, even groot zijn als de kans dat het in de laatste t t.e. van het interval gebeurde. De basiseigenschap van het Poissonproces zegt zelfs dat de kans op een signaal voor elk interval van deze lengte binnen de periode L even groot is. De kans dat de gebeurtenis binnen een bepaald interval van lengte t plaatsgreep, zal dus gelijk zijn aan t/L.

We gaan dit narekenen voor het interval met lengte t aan het begin van het interval van lengte L, we vragen dus naar de kans dat het signaal kwam in de eerste t t.e. De gezochte kans is te schrijven als

$$Pr(N(t)=1 \mid N(L)=1) = \frac{Pr(N(t)=1 \cap N(L)=1)}{Pr(N(L)=1)}$$

De teller rechts is

$$Pr(N(t)=1) \times Pr(N(L-t)=0)$$

Invullen van de kansen voor $N(t)$, $N(L-t)$ en $N(L)$, die Poissonverdeeld zijn, geeft inderdaad

$$Pr(N(t)=1 \mid N(L)=1) = \frac{\lambda t e^{-\lambda t} e^{-\lambda(L-t)}}{\lambda L e^{-\lambda L}} = t/L$$

13.3. VERDERE EIGENSCHAPPEN POISSONPROCES

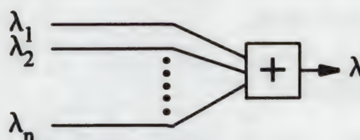
13.3.1. samenstellen en splitsen

De prettige eigenschappen blijven bewaard als Poissonprocessen samensmelten of opgesplitst worden.

Als er verschillende Poissonprocessen met parameters λ_1 tot en met λ_n samen worden genomen, ontstaat er één proces, waarvoor de kans op een signaal in dt t.e. gelijk is aan

$$(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \cdots + \lambda_n)dt$$

(zie figuur 13.2).

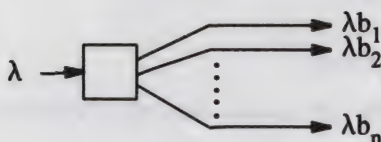


Figuur 13.2. Poissonprocessen vormen samen weer een Poissonproces. De snelheid van optreden van een signaal uit het totale proces is de som van de snelheden van optreden van de deelprocessen.

Het resulterende proces, dat het 'overall' optreden van de signalen beschrijft, is dus ook weer Poissonverdeeld, met als parameter de som van de parameters van de samenstellende processen. De herleidingen voor het gewone 'enkelvoudige' Poissonproces gelden ook voor het samengestelde proces. Het overall optreden van signalen van verschillende soort is zelfs in goede benadering een Poissonproces als de samenstellende processen dat niet zijn, mits er maar veel processen onafhankelijk van elkaar signalen leveren (§12.4).

Als de signalen uit een Poissonproces worden toegeschreven aan deelprocessen (genummerd van 1 tot en met n; figuur 13.3), is elk van die deelprocessen weer een Poissonproces, zolang een optredend signaal de vaste kans b_i heeft om tot het proces met nummer i te worden gerekend. De kans b_i is de kans dat deelproces i de 'beurt' heeft ($\sum_i b_i = 1$).

Er komen ook bij het Poissonproces eenvoudige verdelingen voor, die niet meer het lukrake karakter hebben. Als de signalen bijvoorbeeld consequent om en om aan twee deelprocessen worden toegewezen, zijn de tussenpozen tussen signalen noch voor het deelproces-1, noch voor het deelproces-2 negatief exponentieel verdeeld.



Figuur 13.3. Splitsen van een Poissonproces. Kans b_i dat het deelproces met nummer i optreedt.

De tussenpozen tussen signalen bij een deelproces zijn de som van twee opeenvolgende tussenpozen uit het Poissonproces en dus de som van twee negatief exponentieel verdeelde duren. De som van negatief exponentieel verdeelde grootheden heeft de zogenaamde Erlangverdeling en is niet zonder geheugen. Deelproces-1 en deelproces-2 zijn dan geen Poissonprocessen. Pas als de signalen lukraak verdeeld worden over de twee deelprocessen zijn de deelprocessen volgens de stelling uit de vorige alinea Poissonprocessen. Er blijkt weer dat men dichter bij het Poissonproces is als er minder vast staat.

13.3.2. werklastsplitsing database

Op zuiver lukrake momenten komen bij een database transakties binnen. Deze transakties komen dus volgens een Poissonproces aan. De transakties bestaan gemiddeld voor een kwart uit updates, er zit geen patroon in de volgorde waarin de typen transakties worden aangeboden. Volgens het voorgaande komen de updates voor de database ook volgens een Poissonproces binnen.

Een derde van de transakties maakt gebruik van de file 'employee' (employee is de absolute topper onder de namen van voorbeelden in leerboeken databases). De transakties die deze file gebruiken vormen ook een Poissonproces.

13.4. RENEWAL-PROCESSEN (MATHEMATISCH INTERMEZZO)

Echt mathematisch scherp waren we in het voorgaande hoofdstuk SIGNALEN EN POISSONPROCES niet. Het optreden van signalen met tussenpozen kan soms geformaliseerd worden als een *renewal-proces*. Bij een renewal-proces zijn de tussenpozen onafhankelijk en identiek verdeeld (independent identically distributed), of anders gezegd: de opeenvolgende tussenpozen kunnen steeds gezien worden als onafhankelijke trekkingen uit dezelfde verdeling. Er is in een renewal-proces geen correlatie tussen de lengten van tussenpozen die vlak na elkaar komen. De kans op een sequentie 'kort lang lang' is even groot als de kans op een sequentie 'kort lang kort' (met lang bijvoorbeeld groter dan tweemaal gemiddeld en kort kleiner dan de helft van het gemiddelde of iets dergelijks). Het Poissonproces is een renewal-proces, waarbij de tussenpozen getrokken worden uit de negatief exponentiële verdeling. Dit is een korte en precieze specificatie van het Poissonproces.

Bij een stapsgewijs verloop in de tijd is er bij het Bernouilli-proces geen enkele samenhang tussen de opeenvolgende toestanden, we stonden daar in §9.8 even bij stil. De toestanden zijn in dat eenvoudige geval te zien als trekkingen uit een binomiale verdeling met bepaalde 'kansen op de beurt'. Het renewal-proces is de tegenhanger van het Bernouilli-proces voor een continu verloop in de tijd. De stapsgewijze veranderingen volgens een Markovketen gaan al uit boven de simpele binomiaal verdeelde opvolging uit het Bernouilli-proces. Het renewal-proces is dan ook het eenvoudigste model om tussenpozen 'netjes' te beschrijven.

Een renewal-proces wordt vaak gebruikt om het vervangen van onderdelen te analyseren. De tijd dat een onderdeel funktioneert tussen het moment waarop het nieuw wordt ingezet en het moment waarop het wordt vervangen, is een tussenpoos tussen twee akties van de onderhoudsmonteur; dat is tussen twee 'signalen'. Zulke tussenpozen zijn duidelijk in eerste benadering onafhankelijk. De naamgeving renewal stamt van dit 'vernieuwen' van onderdelen.

In onze beschouwingen hebben we ons niet willen beperken tot renewal-processen, in de praktijk treedt immers wel vaak een correlatie op tussen tussenpozen. De eerder gevonden algemene inzichten (bijvoorbeeld in het hoofdstuk BEURTEN EN KANSEN) zijn geldig voor algemene patronen in het optreden van signalen. Ze kunnen bij renewal-processen exact en precies worden bewezen.

13.5. OPERATIONELE ANALYSE EN POISSONPROCES

De eigenschappen van het Poissonproces zijn zo typisch stochastisch, dat ze niet direkt over te zetten zijn naar een operationele filosofie. Toch zijn de meetbare consequenties van een Poissonproces operationeel weer te geven.

In een Poissonproces treden signalen 'lukraak' op, hoeveel er komen hangt niet van de omstandigheden af, de snelheid van optreden is steeds λ. Voor een Poissonproces is typisch dat de kans op een signaal in het tijdintervalletje dt steeds λdt is, waar dit tijdintervalletje ook ligt. Men stelt het ook vaak zo: de snelheid van optreden λ hangt niet van de tijd af. Men kan nu proberen om onder diverse omstandigheden deze snelheid te meten, geen of weinig variatie in de uitkomsten wijst dan op een Poissonproces.

We gaan er van uit dat moet worden nagegaan hoe de snelheid van aankomst van opdrachten in een computerconfiguratie is; een aankomst van een opdracht is een 'signaal'. Opdrachten gaan soms eerst naar de CPU en komen daar binnen in de tijd dat de CPU bezet is; andere komen als de CPU idle is. Weer anderen komen binnen bij een bezet I/O-apparaat of als er een page fault verwerkt wordt, enzovoort.

Een meetduur valt onder andere uiteen in de tijd $WERK_{bezet}$, waarin de CPU bezet is en de tijd $WERK_{idle}$, waarin de CPU idle is. In die tijd komen er IN opdrachten bij de configuratie binnen, waarvan IN_{werk} bij een bezette CPU en IN_{idle} bij een vrije CPU. Een operationele eis voor het Poissonverdeeld zijn van het aankomstpatroon van opdrachten is nu het ('vrijwel') gelijk zijn van de uit-

komsten van de operationele schatters

$$'\lambda_{bezet} = IN_{bezet} \,/\, WERK_{bezet}$$

en

$$'\lambda_{idle} = IN_{idle} \,/\, WERK_{idle}$$

Dit moet ook gelden als bezet slaat op het bezet zijn van I/O-apparaten. Wanneer het aantal wachtenden in een wachtrij gespecificeerd is, mag het geen verschil uitmaken hoeveel er wachten.

Kortom, algemeen, relateer de snelheid van optreden aan de omstandigheden. Alleen als de snelheid weinig met de omstandigheden varieert kan het proces de eigenschappen van een Poissonproces hebben. De statistiek levert hulpmiddelen om na te gaan dat iets lukraak, dus volgens een Poissonproces, optreedt (§7.6).

Heel vaak wordt het aankomstpatroon in een wachttijdensysteem als een Poissonproces beschreven. De aankomstsnelheid van klanten in een wachttijdensysteem zou kunnen variëren met de omstandigheden, bijvoorbeeld met het aantal klanten in het systeem op het moment van aankomst. Bij een Poissonproces van aankomsten is dit echter niet het geval.

13.6. SAMENVATTING

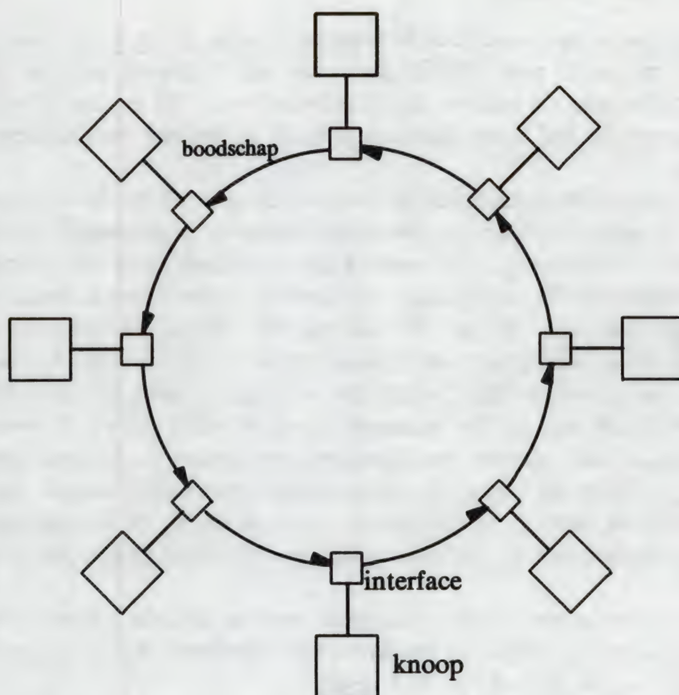
Het Poissonproces draagt, doordat het beknopt de eigenschappen van 'lukraak' en 'zonder samenhang' weergeeft, er belangrijk toe bij dat de elementaire basisrelaties uit de voorgaande hoofdstukken in globale berekeningen kunnen worden toegepast.

13.7. OPGAVEN

De volgende opgave ligt op het gebied van de lokale netwerken. De prestatieanalyse speelt daar tegenwoordig een belangrijke rol. Gevraagd wordt in een eenvoudig geval de doorstroom aan boodschappen in een token-ringnetwerk te berekenen. De vraagstelling kan gemakkelijk naar allerlei kanten gegeneraliseerd worden, zonder dat de aanpak echt anders hoeft te worden.

In een point-to-point ringnetwerk van computers is datacommunicatie in één richting mogelijk. Het net wordt gebruikt voor het berichtenverkeer tussen gebruikers die op de computers —de knopen— zijn aangesloten. Een knoop maakt van een bericht van een gebruiker een of meer boodschappen die door de knoop via de interface op het net worden gezet, zodra daarvoor de gelegenheid is (figuur 13.4).

Elke boodschap is voorzien van het adres van de knoop waarvoor de boodschap bestemd is. Een knoop neemt nota van elke boodschap die hij over de netverbinding van zijn linkerbuurknoop via de interface binnenkrijgt en geeft deze



Figuur 13.4. Tokenring-netwerk. Bij de knopen (groot vierkant) komen boodschappen binnen, die via de interfaces (klein vierkant) op het net gezet worden.

daarna meestal via de interface over de netverbinding aan zijn rechterbuurknoop door. Alleen als hij de boodschap zelf op het net heeft gezet, geeft hij deze niet verder door. Zo'n boodschap is helemaal rond geweest en kan verwijderd worden.

In een ringnetwerk mag nooit meer dan één knoop nieuwe boodschappen op het net zetten, anders zouden er 'botsingen' ontstaan. Dit wordt in een tokenringnetwerk geregeld met behulp van een speciale boodschap, het 'token'. Dit token wordt gebruikt als 'estafette-stokje': de knopen geven elkaar het token door. De knoop die in het bezit is van het token heeft daarmee het recht nieuwe boodschappen in te brengen.

Nieuwe berichten zijn afkomstig van de gebruikers bij een knoop. Ze kunnen meestal niet direkt over het net verzonden worden. Er moet worden gewacht tot de knoop het token van zijn linkerbuurknoop binnenkrijgt. Zodra het token binnen is, worden de nieuwe boodschappen op het net gezet. Na enige tijd kan of mag de knoop geen boodschappen meer inbrengen en geeft hij het token via de interface en de netverbinding door aan zijn rechterbuurknoop. Deze kan op zijn beurt nieuwe boodschappen gaan inbrengen, enzovoort. Een knoop die het token krijgt en geen nieuwe boodschappen heeft, geeft het token zonder meer door.

opgave 13.1: token-ringnetwerk

Er zijn drie knopen A, B en C. Boodschappen worden doorgegeven van A naar B, van B naar C en van C naar A. De duur van het transport van een boodschap van de ene knoop naar de andere wordt verwaarloosd. Er is dus op elk moment precies één knoop die het token ontvangen heeft en nieuwe boodschappen op het net mag zetten.

In knoop A komen boodschappen binnen van de gebruikers die op knoop A zijn aangesloten. Er worden alleen nieuwe boodschappen gegenereerd in de tijd dat het token niet in A aanwezig is (in een tokenloze periode voor A); de boodschappen komen volgens een Poissonproces. Als knoop A het token in bezit heeft worden alle boodschappen die in de voorgaande tokenloze periode zijn binnengekomen, op het net gezet (exhaustive service). Als knoop A geen nieuwe boodschappen te verzenden heeft, duurt het e_A msek. vanaf het moment dat het token bij A binnenkomt tot het moment dat het token knoop A weer verlaat. Wanneer A een of meer nieuwe boodschappen op het net zet is deze tijdsduur d_A msek. ($d_A > e_A$). Of er bij knoop A boodschappen zijn hangt van de duur van de tokenloze periode af, deze is het langst als zowel B als C, de voorgangers van A, één of meer boodschappen op het net hebben gezet. Voor de knopen B en C analoog.

In alle drie knooppunten wordt gemiddeld eens in de tien tokenloze msek. een boodschap gegenereerd. Neem in het komende rekenwerk $e_A = e_B = e_C = 0.5$ msek., $d_A = 5.5$, $d_B = 10.5$, $d_C = 15.5$ msek.

De toestanden waarin het net kan verkeren, worden vastgelegd door de knoop waar het token is en of de voorgaande knopen al dan niet een of meer boodschappen hadden. Zo is bijvoorbeeld:

10C : A heeft boodschap(pen) op het net gezet, B niet; token is bij C.

De drie posities corresponderen met een cyclische permutatie van A, B en C. Op de derde positie staat de naam van de knoop waar het token is, op de voorgaande posities staat een 1, respectievelijk een 0 als de betrokken voorganger wel of geen boodschap(pen) had. Zo is ook

01A : B heeft geen boodschap(pen) op het net gezet, C wel; token is bij A.
11B : C heeft boodschap(pen) op het net gezet, A ook; token is bij B.

- We bekijken de toestandsovergangen vanuit toestand 00C. Hoelang hebben de op C aangesloten gebruikers de tijd gehad om nieuwe boodschappen aan te leveren (1 msek.)? Hoe groot is het aantal gegenereerde boodschappen gemiddeld? Hoe groot is de kans dat er geen enkele boodschap is gegenereerd (0.905)? Bepaal de overgangskans naar toestand 01A (0.095).
- Geef aan welke toestanden aan 00C vooraf kunnen gaan (10B, 00B). Hoelang is de tokenloze periode bij B voor toestand 10B (16)? Hoe groot is de kans om vanuit 10B naar 00C te gaan (0.202)?

- Geef de vergelijking voor de kans in de stationaire fase op het optreden van toestand 00C door het betreffende deel van $\pi=\pi\mathbf{P}$ te nemen ($\pi_{00C} = \pi_{10B}\, \mathbf{P}_{10B,00C} + \pi_{00B}\, \mathbf{P}_{00B,00C}$). Laat zien dat deze evenwichtsrelatie ook volgt uit het 'principe van binnendruppelen is weglekken' uit §11.4-5.
- De stationaire toestandsvector (afgerond) is (.133 .029 .038 .133 .133 .038 .016 .147 .123 .025 .039 .146) voor de toestanden 00A, 01A, 10A, 11A, 00B, 01B, 10B, 11B, 00C, 01C, 10C, 11C. Controleer dat aan de aangegeven evenwichtsrelatie voor 00C wordt voldaan. Beschrijf de betekenis van de elementen uit de toestandsvector.
- Controleer ook de vergelijking voor toestand 10B.
- De toestanden en de overgangen kunnen gegroepeerd worden naar 'token bij A', 'token bij B' en 'token bij C'. Waarom is de som van de kansen op de toestanden met 'token bij A' volgens de stationaire toestandsvector $1/3$?
- Een rondgang van het token wordt een cycle genoemd. In welk deel van de cycles brengt C minstens één nieuwe boodschap in het net? Bepaal dit met de stationaire kansen voor de toestanden met 'token bij A' ($3\,(\pi_{01A} + \pi_{11A})$). Idem met die voor 'token bij B'.
- Bepaal hoelang het token gemiddeld bij knoop C is ($0.5+45\,(\pi_{01A} + \pi_{11A})$). En hoelang gemiddeld bij de knopen A en B?
- Bepaal hoelang een cycle gemiddeld duurt.
- Bepaal de doorstroom aan boodschappen van gebruikers, die op knoop C zijn aangesloten (0.053).
- Controleer dat de totale doorstroom aan boodschappen over het net gelijk is aan $2/10$ per msek.
- Hoe lopen de berekeningen wanneer er ook als de knoop het token in bezit heeft gemiddeld eens in de 10 msek. een nieuwe boodschap gegenereerd wordt? Deze boodschappen worden pas bij het volgende bezoek van het token verstuurd.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

14

Inleiding wachttijdensystemen

14.1. INLEIDING

Een opdracht voor een 'device' uit een configuratie zal vaak vertraging oplopen doordat andere opdrachten het device bezig houden. De gemiddelde tijd die gewacht wordt kan in de praktijk vaak met de basisrelaties uit de voorgaande hoofdstukken worden berekend. Dit is bijvoorbeeld het geval als opdrachten op volledig lukrake momenten binnenkomen en de volgorde waarin ze geholpen worden niet van hun serviceduur afhangt.

In de volgende hoofdstukken gaan we dit resultaat bespreken. We betreden daarmee het terrein van de *wachttijdentheorie*; dit hoofdstuk bevat een inleiding daarin.

De prestaties van een wachttijdensysteem worden globaal beschreven door een beperkt aantal grootheden. Tussen *doorstroom* en *bezettingsgraad* en de gemiddelden van *bedienduur*, *serviceduur*, *verblijfduur*, *wachtduur* en het gemiddeld aantal *aanwezigen* en *wachtenden* bestaan een aantal basisrelaties, die samenhangen via de relatie van Little. Voor de servers uit het wachttijdensysteem is van belang of ze steeds met dezelfde snelheid werken *(fixed-rate servers)* of dat hun snelheid van de omstandigheden afhangt *(load-dependent servers)*. Combinaties van wachttijdensystemen kunnen *open* of *gesloten* zijn. Bij open systemen komen er van buiten af klanten binnen, die later weer vertrekken, bij gesloten systemen pendelen klanten voortdurend heen en weer tussen de servers.

14.2. WACHTTIJDENTHEORIE

In de voorgaande hoofdstukken kwamen we telkens situaties tegen, waarin gewacht wordt.

De *wachttijdentheorie*, *wachtrijtheorie* of *queuing theory* houdt zich bezig met het onderzoek van zulke situaties. Deze theorie heeft een rijke historie. Het onderzoek aan wachtrijen is zo'n eeuw geleden begonnen met pionierswerk van de Deen A.K. ERLANG over de wachttijden in telefooncentrales. Er is op dit gebied veel verreikend en diepgravend onderzoek gedaan, wat heel wat resultaten en vuistregels (stem des volks stellingen) heeft opgeleverd, die bij praktische problemen steeds weer nuttig blijken te zijn.

In de loop van de jaren is er in de wachttijdentheorie een beperkt jargon ontstaan, wat we in de voorgaande hoofdstukken in voorkomende gevallen hebben gebruikt. De gebruikelijke standaard-nomenclatuur is lang geleden ontleend aan een plaats, die iedereen nog altijd heel goed kent: het wachthokje voor een loket (figuur 14.1).



Figuur 14.1. Wachttijdensysteem met 1 server. Een server wordt gestileerd als een cirkel. De wachtrij (queue) wordt getekend als een wachthokje.

Als ergens gewacht wordt zijn er objecten die geassocieerd zijn met een of andere verwerkingsopdracht. Ze worden formeel altijd *klanten* met *opdrachten* genoemd. De opdrachten worden verwerkt door een processor, of een terminaldriver, een randprocessor, enzovoort. Dit is de *server*, of *bediende* (maar dan dreigt verwarring: bedient de bediende of wordt de bediende bediend? Daarom gebruiken we liever het onbelaste woord server). De server bedient of verwerkt klanten. Er is een *beperkte wachtruimte* als er slechts een beperkt aantal wachtende klanten kan zijn. Het geheel van een of meer servers en hun klanten is het *wachttijdensysteem*.

In concrete praktijksituaties kan soms alleen al het indelen volgens een wachttijdensysteem verhelderend zijn. Zo is het altijd zinvol aan te geven wie in een bepaalde situatie de server is en wie de klanten zijn. De CPU bijvoorbeeld is voor de uit te voeren machine-instrukties de server, maar CPU's kunnen ook klanten zijn, die wachten op toegang tot een bepaald stuk beschermde code.

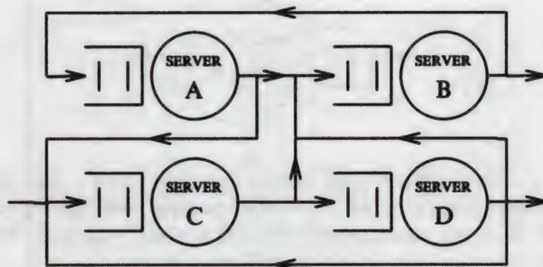
14.3. NOMENCLATUUR

Klanten worden tijdens hun gang door een systeem door een of meer servers geholpen. Wanneer ze meer dan één server aandoen, kan het zijn dat ze eerst de ene server aandoen, dan de andere, dan de volgende, enzovoort, zonder ooit terug te keren bij een server waar ze al eens geweest zijn. Ze worden dan door een *tandem van servers* verwerkt (figuur 14.2).



Figuur 14.2. Combinatie van wachttijdensystemen: tandem van servers. Er komen bij A klanten binnen. Klanten gaan van A naar B en van B naar elders.

Het is ook mogelijk dat ze een tijd lang heen en weer pendelen tussen een aantal servers en dus afwisselend door deze servers geholpen worden.

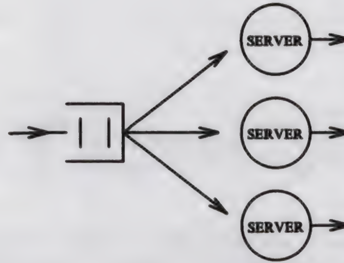


Figuur 14.3. Queuing-netwerk, combinatie van de wachttijdensystemen bij A, B, C en D. Er komen bij C klanten van buiten binnen. Klanten gaan van A naar B en C, van B naar A, van C naar B en D, van D naar B en C en er vertrekken klanten naar elders na verwerking door B en D.

Ze worden dan door een *queuing-netwerk* verwerkt (figuur 14.3, figuur 14.7 uit §14.8 en figuur 14.9 uit §14.9). Verwerking door een tandem van servers doet zich onder andere voor wanneer een bericht in een computernetwerk van knoop naar knoop getransporteerd wordt. Een queuing-netwerk treedt standaard op bij de verwerking in een computerconfiguratie (vergelijk §6.2 en §9.9).

Bij iedere server uit een queuing-netwerk of uit een tandem hoort een apart wachttijdensysteem. De aanvoer van klanten bij dit systeem komt van buiten of vanaf de andere servers. Zowel een queuing-netwerk als een tandem van servers is

dus een combinatie van wachttijdensystemen. In de wachttijdentheorie spreekt men alleen dan van een 'wachttijdensysteem met meerdere servers' als elke klant van het wachttijdensysteem door maar één en niet meer dan één van de servers wordt geholpen.



Figuur 14.4. Wachttijdensysteem met 3 servers.



Figuur 14.5. Wachttijdensysteem met 1 aggregate-server. Een aggregate-server ontstaat bijvoorbeeld als de tandem van twee wachttijdensystemen uit figuur 14.2, of de vier wachttijdensystemen A, B, C en D uit figuur 14.3, of de drie servers uit figuur 14.4, als een geheel worden gezien.

Elk queuing-netwerk is op een 'hoger' niveau in principe te zien als een enkele *samengestelde* (dat is 'aggregate') server (figuur 14.5). Het queuing-netwerk kan op dat niveau vervangen worden door een wachttijdensysteem rond deze *aggregate-server*. De eigenschappen van de aggregate-server volgen uit die van de samenstellende servers. Ook een tandem van servers of de servers uit een wachttijdensysteem met meerdere servers kunnen als een enkele aggregate-server worden gemodelleerd.

14.4. BASISGROOTHEDEN WACHTTIJDENSYSTEMEN

14.4.1. inleiding

Zoals steeds zullen we alleen maar letten op de stationaire fase, waarin het wachttijdensysteem gemiddeld genomen stabiel is; de uitspraken gelden 'op de lange termijn'. Op de lange termijn betekent bij Markovketens 'voor grote *n*', de tijd verloopt stapsgewijs (§9.3). Op de lange termijn is voor de echte continue 'kloktijd', waarin we nu geïnteresseerd zijn, 'voor na een lange tijdsduur'. In beide gevallen is 'op de lange termijn' ook 'over de lange termijn'.

Er wordt over de stationaire fase informatie verzameld met operationele schatters, die gemeten worden in representatieve meetduren. Het detail waarin men het funktioneren kan en wil beschrijven wordt bepaald door het detail waarmee men gegevens kan en wil verzamelen. In principe kan erg veel informatie bijgehouden worden. Bij wachttijdensystemen gaat het in eerste instantie om informatie over een aantal basisgrootheden die werklast en drukte globaal beschrijven. Deze basisgrootheden gaan we nu vastleggen.

14.4.2. doorstroom

We gaan er gemakshalve van uit dat er één server is. Deze ene server zal vaak een aggregate-server zijn, dus een geheel van (deel)servers.

In de tijdsduur *MEETDUUR* (de meettijd) treden *IN* klanten binnen en is de server gedurende *WERK* t.e. bezig met het verwerken van opdrachten van klanten (met 'hulp voor klanten'). Er verlaten in de tijdsduur *MEETDUUR* een aantal *UIT* klanten het systeem omdat ze geholpen zijn.

De meetduur *MEETDUUR* moet betrekkelijk lang worden genomen, anders zullen de waarnemingen niet relevant zijn. Er geldt dan vrijwel

$$IN = UIT$$

We mogen dit zeggen, omdat we er van uitgaan dat klanten die het systeem ingaan er ooit geholpen uitkomen, er verdwijnen geen klanten. Elk krijgt een beurt binnen een tijdsduur die kort is vergeleken met *MEETDUUR*. De bedieningsregeling zal 'fair' zijn: elke klant komt ooit aan bod in eindige tijd, er is geen 'starvation' van (groepen) klanten. De relatie $IN = UIT$ is een operationele 'behoudsrelatie' als in §11.2 en beschrijft het 'behoud van klanten'. Zo'n relatie komt men in de stationaire fase altijd tegen, we zagen dit al bij de Markovketens; in het hoofdstuk MARKOVKETENS EN EVENWICHTSVERGELIJKINGEN plaatsten we enkele kanttekeningen.

Het aantal klanten dat per tijdseenheid door de server geholpen wordt noemen we de doorstroom van het systeem. Dit is hetzelfde begrip als het vertrouwde 'aantal opdrachten dat door de configuratie per tijdseenheid verwerkt wordt' (de klanten zijn daarbij de opdrachten, de server is de configuratie, §5.2).

De operationele schatter ervoor is

$$'doorstroom = UIT\,/\,MEETDUUR$$

en volgens behoud van klanten dus ook

$$'doorstroom = IN\,/\,MEETDUUR$$

Dit laatste is eigenlijk de *instroom* of de *snelheid van aankomst of binnenkomst*. Deze wordt geschat via

$$'(overall)\ snelheid\ van\ binnenkomst = IN\,/\,MEETDUUR$$

In een stationair wachttijdensysteem is de doorstroom gelijk aan de 'uitstroom' en gelijk aan de 'instroom'. Het binnenkomen van een klant is in de termen van het hoofdstuk SIGNALEN EN POISSONPROCES het optreden van het signaal, dat er een nieuwe klant is. De doorstroom is, dank zij het behoud van klanten en dank zij de stationaire fase, ook de 'overall' snelheid van optreden van deze signalen (§12.2, §13.5). In de literatuur over wachttijdensystemen wordt meestal de voorkeur gegeven aan de benaming 'arrival rate' boven doorstroom, daar wordt dus de 'snelheid van aankomst' als basisgrootheid gekozen. Wij houden vast aan 'doorstroom', omdat dit aansluit bij de standaard-terminologie uit de computer-prestatieanalyse.

14.4.3. bedienduur

Naast de doorstroom voeren we als basisbegrip de *gemiddelde bedien(ings)duur* in. Deze wordt geschat via

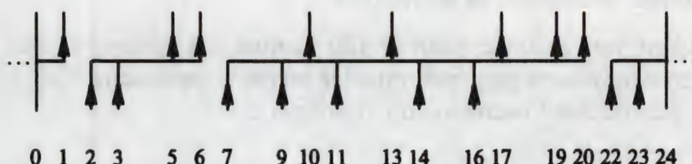
$$'bedienduur = WERK\,/\,UIT$$

De grootheid *WERK / UIT* kan gezien worden als de gemeten gemiddelde tussenpoos bij het verlaten van het systeem, genomen in *virtual time*, dat is bij werkende server. Het is de gemiddelde 'time between completions'. Tussen twee opeenvolgende momenten waarop een klant vertrekt, zit gemiddeld deze tijdsduur, tijden van leegloop niet meegenomen in de telling (figuur 14.6). Met 'in virtual time' wordt algemeen bedoeld: volgens het gewone verloop van de tijd, maar met de tijd die niet interessant is weggelaten.

De gemiddelde bedienduur is dus zo gedefinieerd dat

$$'bedienduur = {}'tussenpoos\ uittreden\ \text{(virtual time)}$$

Als de server werkt levert hij 'om de bedienduur' een klant af, de gemiddelde



Figuur 14.6. Aankomst (pijlen naar de tijdlijn toe) en vertrek (pijlen van de tijdlijn af) in een wachttijdensysteem. Tijdsverloop van links naar rechts. De *MEETDUUR* loopt van 0 tot 24, *WERK* = 20, *IN* = 9, *UIT* = 8. Gemeten gemiddelde bedienduur = 20/8. Gemeten instroom is 9/24, gemeten doorstroom (of uitstroom) = 8/24, gemeten bedienduren 3, 1, 3, 3, 4, 2, 1. Busy periods van 2-6 en van 7-20. Tijdsduren in tijdseenheden, doorstromen per tijdseenheid.

bedienduur is de tijd die de server gemiddeld per klant bezig is. De 'effektieve verwerkingssnelheid' is één klant per gemiddelde bedienduur.

In het hoofdstuk SIGNALEN EN POISSONPROCES is benadrukt dat naast de 'snelheid van optreden' altijd de gemiddelde tussenpoos staat. De gemiddelde tussenpoos bij aankomst wordt geschat via

$$\text{'}\textit{tussenpoos aankomst} = \textit{MEETDUUR} \,/\, \textit{IN}$$

De aangegeven operationele schatters zijn 'eventueel' nette schatters voor de verwachtingswaarden van overeenkomstige stochasten (§4.4). Met operationele schatters zullen immers —als het goed is— verwachtingswaarden van stochasten corresponderen. De overeenkomst van een operationele schatter en een verwachtingswaarde geven we aan door het teken $\rightarrow$ (als in §5.9). De verbanden zijn hier (*a* van aankomst, *u* van uittreden)

$$\text{'}\textit{tussenpoos aankomst} \rightarrow E(T_a)$$

$$\text{'}\textit{bedienduur} = \text{'}\textit{tussenpoos uittreden} \rightarrow E(T_u)$$

met $E(T_a)$ als de 'echte' gemiddelde tussenpoos bij aankomst en $E(T_u)$ als de 'echte' gemiddelde tussenpoos bij het verlaten van de aktieve server.

Er wordt in de wachttijdentheorie soms ook gelet op klanten die niet binnenkomen, zulke klanten zijn van plan in te treden maar ze worden afgeschrikt, of komen niet binnen omdat de wachtruimte vol zit. We volgen hier een 'nononsense' aanpak: zulke klanten bestaan voor ons niet. Er hoeven daardoor geen wachtduren te worden gemiddeld† over klanten die wel of niet worden verwerkt.

---

† Er is daarom geen onderscheid tussen aankomst en binnenkomst.

14.4.4. voorbeeld: bedienduur en doorstroom

In een meetduur van 1000 t.e. gaan er 100 klanten het systeem binnen en komen er 110 klanten het systeem geholpen uit. De server is gedurende 770 t.e. bezig. De gemeten gemiddelde tussenpoos bij aankomst is

$$'\textit{tussenpoos aankomst} = 1000/100 = 10 \text{ t.e.}$$

De gemeten gemiddelde instroom is

$$'\textit{instroom} = 100/1000 = 0.10 \text{ klanten per t.e.}$$

De gemeten gemiddelde bedienduur is

$$'\textit{bedienduur} = 770/110 = 7 \text{ t.e.}$$

Dit is ook de gemiddelde tussenpoos tussen twee momenten van vertrek als leegloop niet als relevante tijd wordt gezien (in 'virtual time'). Als die tijd wel wordt meegenomen is

$$'\textit{tussenpoos uittreden} \text{ (kloktijd)} = 1000/110 = 9.1 \text{ t.e.}$$

De gemeten gemiddelde doorstroom is

$$'\textit{doorstroom} = 110/1000 = 0.11 \text{ klanten per t.e.}$$

Deze is vrijwel gelijk aan de gemiddelde instroom of 'arrival rate' van 0.10 klanten per t.e. Maar hij verschilt wezenlijk van de (bedien)snelheid van 1/7 klanten per t.e., waarmee de server werkt als er werk voor hem is.

14.4.5. bezettingsgraad

De server zal niet de hele meetduur *MEETDUUR* gewerkt hebben. Hij heeft tijdens $MEETDUUR - WERK$ niets te doen gehad en heeft maar gedurende het deel $WERK / MEETDUUR$ van zijn tijd gewerkt. Een op een willekeurig moment binnenkomende klant zal met een kans U de server werkend aantreffen, met een kans $1 - U$ vindt hij de server zonder werk (idle). De operationele schatter voor de *bezettingsgraad* U is

$$'\textit{bezettingsgraad} = WERK / MEETDUUR$$

Bij de bezettingsgraad wordt in de operationele schatter gemiddeld over systematische veranderingen in de loop van de tijd. Net als bij de doorstroom is het niet gebruikelijk deze middeling expliciet in de naamgeving op te nemen (§5.2). Het begrip 'bezettingsgraad' is in de voorgaande hoofdstukken vaak gebruikt (§4.6.2, §7.2.4, §7.5.2). De bezettingsgraad van de server wordt ook de utilisatie van de server genoemd (vandaar de *U*) met

$$U = utilisatie \leftarrow \text{'}bezettingsgraad$$

14.4.6. serviceduur

Een server uit een computerconfiguratie zal in het algemeen niet onder alle omstandigheden even snel werken. Het is geen *fixed-rate* server met een vaste verwerkingssnelheid, maar een *load-dependent* server met een verwerkingssnelheid, die van de omstandigheden afhangt† (we gebruiken de benamingen uit de literatuur).

De volgende servers zijn bijvoorbeeld load-dependent:
Een CPU krijgt als het druk wordt steeds meer extra werk per opdracht voor het beheer van de verwerking in de configuratie, deze 'overhead' maakt dat hij trager wordt (§6.2.2).
Bij een disk gaat de kop bij de volgende seek naar de dichtstbijgelegen cylinder waarvoor een seekopdracht is geplaatst (SHORTEST SEEK TIME FIRST). Als het druk is, zijn er veel wachtende opdrachten en is de seek gemiddeld korter, dus de verwerking per opdracht sneller dan wanneer er bijna nooit een wachtende klant is.
Drie devices kunnen met een eigen vaste snelheid een bepaald soort opdrachten uitvoeren (als in figuur 14.4). Als het druk is werken de devices meestal tegelijk en in een slappe tijd werkt er af en toe één. Als hun snelheden gelijk zijn werken ze als het druk is ongeveer driemaal zo snel als wanneer er weinig te doen is. Wanneer de drie servers samen als een aggregate-server worden gezien, is deze server load-dependent.

Het is daarom heel vaak niet mogelijk van te voren te zeggen hoelang de verwerking van een klant door de server zal duren. Toch moet er een maat zijn voor de hoeveelheid werk, die elke klant aanlevert. Die vinden we door bij iedere klant op te geven hoelang zijn verwerking zou duren op een *standaardserver* onder *standaard* omstandigheden. De duur van de verwerking op de standaardserver is de *serviceduur*. Zo'n standaardserver is meestal gemakkelijk te vinden.

Bij de voorbeelden zijn de standaardservers als volgt. Bij de verwerking door de CPU is het de CPU zelf, maar dan in een systeem zonder overhead. De serviceduur is de CPU-tijd van de opdracht wanneer er geen overhead is. Bij de disk

---

† Load slaat op werklast, de verwerkingssnelheid varieert met de totale last (load), die de server moet verwerken.

wordt gestandaardiseerd op een opdracht met een gegeven seektijd. Bij de drie devices is de serviceduur van een opdracht de verwerkingstijd op een van de drie servers, die als standaard wordt gekozen.

Soms wordt de hoeveelheid werk uitgedrukt in bits (bij transport over een lijn) of in instrukties (voor een processor). Dit kan altijd worden vertaald naar een tijdsduur en dus een serviceduur. Wanneer bijvoorbeeld een lijn met een snelheid van 30Kbits/sek. als standaardserver wordt gekozen, betekent een opdracht van 300 bits een serviceduur van 10 msek.

Als de server steeds met dezelfde snelheid werkt, kan de server zelf als de standaardserver worden gezien, omdat de verwerkingssnelheid altijd dezelfde is, wat de omstandigheden ook zijn. Bij een fixed-rate server is de serviceduur van een opdracht daarmee gewoon de verwerkingstijd.

De werklast die een server moet verwerken, wordt voortaan gekarakteriseerd door de verdeling van de servicedurenâ€ (§2.2.3, §9.9.2). De stochast T_s zal de serviceduur aangeven; de gemiddelde serviceduur is $E(T_s)$.

Bij een fixed-rate server is de gemiddelde serviceduur (met de server zelf als standaardserver) gelijk aan de gemiddelde bedienduur. Dit kan met operationele schatters expliciet worden gemaakt.

Als de server in de *MEETDUUR* gedurende *WERK* t.e. werkt, is vrijwel (afgezien van randeffekten; *i* loopt over de UIT in die tijd afgewerkte opdrachten)

$$\sum_{i=1}^{UIT} serviceduur_i = WERK$$

De server is als standaardserver immers voor een klant met een serviceduur $serviceduur_i$ precies $serviceduur_i$ t.e. bezig. De gemiddelde serviceduur wordt geschat met

$$gemiddelde\ serviceduur = \sum_{i=1}^{UIT} serviceduur_i \,/\, UIT \approx WERK \,/\, UIT$$

terwijl volgens afspraak ook

$$'bedienduur = WERK \,/\, UIT$$

De gemiddelde waarden van bedienduur en serviceduur zijn voor servers met een

---

† In de literatuur worden voor zowel serviceduur als bedienduur benamingen gebruikt als service time, service demand, load demand, enzovoort.

vaste verwerkingssnelheid dus gelijk:

$$E(T_u) = E(T_s)$$

De verdelingen van de serviceduren en de bedienduren hoeven natuurlijk niet hetzelfde te zijn. Die kunnen in principe sterk verschillen en zullen dat ook vaak doen.

Vanuit de klant gezien ligt de beschrijving van een wachttijdensysteem zo: hij brengt een hoeveelheid werk ter grootte van de serviceduur in en blijft in het wachttijdensysteem totdat dit werk verwerkt is. In die tijd wacht hij op verwerking en wordt hij verwerkt. Maar vanuit de server gezien is het anders: hij brengt het deel 'bezettingsgraad' van zijn tijd werkend door en verwerkt in die tijd gemiddeld 'om de bedienduur' een klant.

Als de server niet altijd met dezelfde snelheid werkt maar load-dependent is, lopen de gemiddelde serviceduur en de gemiddelde bedienduur uiteen. De gemiddelde serviceduur is per definitie onafhankelijk van de omstandigheden, maar de gemiddelde bedienduur varieert bij een load-dependent server met de omstandigheden en zal verschillen naar gelang de drukte. In de inverse van de gemiddelde bedienduur manifesteert zich de snelheid waarmee de server onder de huidige omstandigheden werkt, het is de 'effektieve verwerkingssnelheid'.

Het verschil tussen bedienduur en serviceduur hangt samen met een verschil in visie op het wachttijdensysteem: gaat het om collectieve of individuele grootheden. De serviceduur is een individuele grootheid met een verdeling per groep klanten en specificeert de werklast. De bedienduur is een collectieve grootheid, die aangeeft met welke snelheid deze werklast onder de heersende omstandigheden (zoals afwikkelingsvolgorde en drukte) verwerkt wordt.

Het verschil tussen een load-dependent server en een fixed-rate server is voor het praktisch analyseren van de prestaties van configuraties erg belangrijk. Een load-dependent server zal zich gelukkig vaak voor een bepaald traject van de hem opgelegde belasting als fixed-rate server voordoen. Voor die belastingen is het een server, die steeds even snel werkt. Dit is bijvoorbeeld het geval bij de load-dependent server 'diskserver' uit §8.9.

In het volgende hoofdstuk beperken we ons tot servers met een vaste verwerkingssnelheid. De volgorde van verwerken zal bovendien zo zijn dat een bedienduur direkt correspondeert met een serviceduur. Voor deze 'klassieke' wachttijdensystemen ligt de zaak eenvoudig.

De uitkomsten van deze klassieke wachttijdensystemen worden soms op servers toegepast die in het te analyseren gebied sterk load-dependent zijn, de uitspraken die dan gedaan worden zijn onbetrouwbaar. Dit is een van de valkuilen van de prestatie-analyse.

14.5. BASISRELATIE DOORSTROOM, BEZETTINGSGRAAD, BEDIENDUUR

Tussen 'doorstroom, 'bedienduur en 'bezettingsgraad bestaat een fundamentele samenhang. Eenvoudig invullen laat zien dat

$$
\begin{aligned}
\text{'}doorstroom &= UIT \,/\, MEETDUUR \\
&= (WERK \,/\, MEETDUUR) \,/\, (WERK \,/\, UIT)
\end{aligned}
$$

of

$$\text{'}doorstroom = \frac{\text{'}bezettingsgraad}{\text{'}bedienduur}$$

We hebben hier kennelijk een belangrijke en heel nuttige consequentie van de afspraken voor het schatten in handen. Door de splitsing van de tijdsduur *MEETDUUR* in een tijd van werken *WERK* en een tijd van nietsdoen *MEETDUUR* − *WERK* wordt ook de doorstroom gesplitst in een produkt van twee factoren. De teller geeft de kans dat er werk voor de server is, de noemer beschrijft hoe snel de server effektief werkt als er werk is. De relatie is fundamenteel en ligt gelukkig voor de hand. In de beperkte vorm is het een identiteit tussen operationele schatters, maar de uitgebreide vorm is een relatie tussen de 'echte' gemiddelde bedienduur, doorstroom en bezettingsgraad (§4.4). De vorm voor de 'echte' waarden is

$$doorstroom = \frac{bezettingsgraad}{gemiddelde\ bedienduur}$$

waarin voor gemiddelde de verwachtingswaarde moet worden gelezen, dus

$$1 \,/\, E(T_a) = U \,/\, E(T_u)$$

We noemen de betrekking de basisrelatie tussen de doorstroom, de bezettingsgraad en de gemiddelde bedienduur.

Het is een kenmerk van de ware basisrelatie dat hij wordt gebruikt zonder te worden genoemd. De hier geformuleerde betrekking is bijvoorbeeld in §9.9 op meerdere plaatsen toegepast. Het is niet strikt nodig de relatie uit te spreken, maar door het te doen worden redeneringen helderder.

Bij een server met een vaste snelheid is de basisrelatie gewoon

$$doorstroom = \frac{bezettingsgraad}{gemiddelde\ serviceduur}$$

Omdat de bezettingsgraad nooit hoger dan 1 kan zijn, geldt daarmee

$$\textit{doorstroom} \leqslant (\textit{gemiddelde serviceduur})^{-1}$$

Bij een fixed-rate server is er dus een bovengrens voor de doorstroom.

Ook deze elementaire afgrenzing is in de voorgaande hoofdstukken al gebruikt, bijvoorbeeld in §6.2 en §9.9 (§14.9.3). Als de instroom hoger is dan 'één per gemiddelde serviceduur' treedt er geen evenwicht meer op en groeit de rij wachtenden maar door. Het systeem komt dan niet meer in stationaire fase, er is alleen maar een 'transient state' (§9.6.1). Voor die omstandigheden zullen we wachttijdensystemen niet bekijken.

14.5.1. vervolg voorbeeld: bedienduur en doorstroom (14.4.4)

De gemeten bezettingsgraad is

$$\textit{'bezettingsgraad} = 770/1000 = 0.77$$

Inderdaad geldt voor de gemeten waarden van de operationele schatters voor de doorstroom, de bezettingsgraad en de gemiddelde bedienduur de basisrelatie

$$0.11 = 0.77/7$$

14.5.2. voorbeeld: lijnberichten

Over een lijn worden berichten verstuurd met transfertijden, die liggen tussen 9 en 18 msek. De verdeling van de transfertijden heeft een variatiecoëfficiënt van 0.5, de gemiddelde transfertijd is 15 msek. Na het versturen van het eerste bit duurt het tussen 20 en 30 msek. voordat het volgende bericht wordt aangeboden, gemiddeld duurt dit 25 msek. Een nieuw bericht vindt de lijn dus vrij en kan altijd direkt overgestuurd worden.

Er komt gemiddeld om de 25 msek. een opdracht, de doorstroom is dus 1000/25 = 40 berichten per sekonde. De bezettingsgraad van de server 'lijn' is volgens de basisrelatie $40 \times 0.015 = 0.60$ (of 15/25, het is ook de verhouding van de gemiddelde tussenpozen bij vertrek en bij aankomst).

14.6. VOORBEELD: DRIE SERVERS SAMEN

Een oppervlakkig voorbeeld kan het verschil tussen serviceduur en bedienduur illustreren.

Een wachttijdensysteem bestaat uit drie servers, die alle drie met een vaste snelheid werken (figuur 14.4). Ze verwerken opdrachten, die door elk van hen in

gemiddeld 10 msek. verwerkt kunnen worden; de servers zijn even snel. De gemiddelde serviceduur van de opdrachten is als we één van hen als standaardserver zien, 10 msek.

Veronderstel eerst dat alle drie servers het erg druk hebben. Iedere server is bijna continu bezig en zal vrijwel om de 10 msek. een afgewerkte opdracht afleveren. Door de drie servers samen worden zo gemiddeld per 10 msek. bijna 3 opdrachten afgeleverd. De doorstroom aan opdrachten is bij grote drukte dus nagenoeg 300 opdrachten per sekonde.

De drie servers kunnen samen als een enkele aggregate-server worden gezien. De bezettingsgraad van deze server is bij grote drukte bijna 1. De basisrelatie tussen doorstroom, bezettingsgraad en gemiddelde bedienduur zegt dat de gemiddelde bedienduur bij de aggregate-server vrijwel 10/3 msek. is, er vertrekt gemiddeld om de 10/3 msek. een klant.

Als het helemaal niet druk is zal er slechts af en toe een server werk hebben. Het komt bijna niet voor dat er twee servers tegelijk werken. De bezettingsgraad van de aggregate-server is dan vrijwel gelijk aan de som van de bezettingsgraden U_i van de individuele servers, omdat meestal òf de een òf de ander bezet is. Dus $U_{aggr} = \sum U_i$. De doorstroom door het geheel is altijd gelijk aan de som van de doorstromen door de servers apart. Omdat bij iedere server geldt dat $doorstroom_i = U_i / 10$ (per msek.) geldt ook dat

$$totale\ doorstroom = \sum_i U_i / 10$$

Bij geringe drukte is de doorstroom door de aggregate-server dus

$$doorstroom\ door\ aggregate\text{-}server = U_{aggr} / 10$$

waaruit ook voor de twijfelaars blijkt dat de gemiddelde bedienduur dan vrijwel 10 msek. is.

Het geheel van de drie servers gedraagt zich als een load-dependent server. Bij geringe drukte doet het systeem zich voor als een server met de gemiddelde serviceduur als gemiddelde bedienduur. Maar bij grote drukte is de gemiddelde bedienduur nagenoeg een derde van de gemiddelde serviceduur; de effektieve snelheid is dan bijna driemaal zo groot als bij geringe drukte. De server is load-dependent doordat er bij grotere drukte gemiddeld meer servers parallel werken.

De drie servers kunnen verschillende snelheden hebben. Veronderstel dat van de drie servers er twee veel trager zijn dan de derde: de servers B en C doen gemiddeld tweemaal zo lang over een opdracht als server A. De verwerkingstijd van een opdracht op server A blijft gemiddeld 10 msek., maar B en C doen over een opdracht gemiddeld 20 msek. De gemiddelde serviceduur van een opdracht is ook nu 10 msek., tenminste als A als standaardserver wordt gekozen. Met B als standaardserver zou de gemiddelde serviceduur 20 msek. zijn. De doorstromen

door A, B en C zijn respectievelijk $U_A/10$, $U_B/20$ en $U_C/20$ opdrachten per msek. Als het erg druk wordt groeit de doorstroom tot maximaal (bijna) $1/10 + 1/20 + 1/20$ opdrachten per msek. (dit geldt ook als B en C pas traag worden als ze het druk krijgen, ze zijn bijvoorbeeld bij weinig werk even snel als A maar bij topdrukte werken ze op halve snelheid). De bijbehorende gemiddelde bedienduur is $1/(1/10+1/20+1/20) = 5$ msek.

De gemiddelde bedienduur voldoet volgens de basisrelatie als het helemaal niet druk is aan

$$\textit{gem. bedienduur} = (U_A + U_B + U_C)/(U_A/10 + U_B/20 + U_C/20)$$

maar hieruit kan nu niet zonder verdere kennis over de U's de gemiddelde bedienduur worden bepaald. Omdat informatie over de scheduling ontbreekt is het niet mogelijk aan te geven hoe de klanten over de servers verdeeld worden.

Bij een load-dependent server is het in principe pas mogelijk de gemiddelde bedienduur aan te geven als het systeem na specificatie helemaal is 'doorgerekend', bijvoorbeeld via evenwichtsvergelijkingen als in §16.2 of volgens de methoden uit §14.9.2. De gemiddelde bedienduur is geen invoerparameter zoals de gemiddelde serviceduur.

14.7. BASISRELATIES WACHTTIJDENSYSTEMEN

14.7.1. inleiding

Er bestaan een aantal elementaire verbanden, die voor elk wachttijdensysteem geldig zijn. Enkele daarvan kennen we al uit §4.5.1, het zijn de relaties van Little. In §14.5 formuleerden we de basisrelatie tussen de doorstroom, de bezettingsgraad en de gemiddelde bedienduur. De andere volgen nu in hun meest voorkomende vorm.

Bij wat bizarre wachttijdensystemen is het altijd zaak de formules niet blindelings toe te passen, maar steeds kort de 'afleiding' langs te lopen. Door dat te doen kan de gedachtengang voor het speciale geval gereconstrueerd worden en zijn afwijkingen direkt te herkennen en aan te vullen. Dit voorkomt het ontstaan van 'paradoxen'.

14.7.2. Little voor wachttijdensystemen

We gaan terug naar een van onze allereerste inzichten: de relatie van Little. Het verblijf in een wachttijdensysteem wordt daarvoor in termen van stochasten beschreven†.

---

† Een in de literatuur veel gebruikte notatie staat in §4.5.1.

De tijd, die in het systeem wordt doorgebracht, noemen we de *verblijfduur*, de gemiddelde verblijfduur is E(*verblijfduur*). De verblijfduur wordt ook wel de *responsetijd* genoemd (zoals in §8.9). De tijd die gewacht wordt is de *wachtduur*, met E(*wachtduur*) als de gemiddelde wachtduur. De operationele schatters voor de gemiddelde verblijfduur en wachtduur zijn 'verblijfduur en 'wachtduur.

Het gemiddeld aantal klanten, dat op een *willekeurig* moment in het systeem aanwezig is, wordt aangegeven als het gemiddeld aantal *aanwezigen* (dit wordt in de praktijk vaak de gemiddelde *queuelengte* genoemd, de benaming van prestatiegrootheden loopt sterk uiteen). Het betreft het tijdgemiddeld aantal, dat is ook het gemiddeld aantal 'op de lange termijn' en 'over de lange termijn', dus het gemiddeld aantal in de stationaire fase (de redeneringen bij Markovketens voor een stapsgewijs tijdsverloop worden gegeneraliseerd naar een continu tijdsverloop, waarin niet de stap n, maar het tijdstip t het verloop parametriseert). Het is ook het aantal dat een lukrake waarnemer gemiddeld zal vinden. De operationele schatters zijn als in §4.2.

Tot de aanwezigen behoren zowel de wachtende klanten als de klant die bij de server in verwerking is (in een systeem met meerdere servers de klanten bij de servers). Naast het gemiddeld aantal 'aanwezigen' onderscheidt men vaak het gemiddeld aantal *wachtenden*, dat is het gemiddeld aantal klanten dat op een willekeurig moment staat te wachten.

De gemiddelde verblijfduur E(*verblijfduur*) en het gemiddeld aantal aanwezigen voldoen volledig aan de beschrijving van 'gemiddelde duur' en 'gemiddeld aantal' uit de relatie van Little (§4.5.1, §7.8). Als 'stroom' moet de doorstroom worden genomen. Er geldt dan ook altoos

$$\textit{gemiddeld aantal aanwezigen} = \textit{doorstroom} \times E(\textit{verblijfduur})$$

Hetzelfde geldt rond 'wachten', weer met doorstroom als 'stroom':

$$\textit{gemiddeld aantal wachtenden} = \textit{doorstroom} \times E(\textit{wachtduur})$$

Beide relaties zijn de eigenlijke 'relaties van Little'. Het zijn vormen van de 'Stelling van Little', die door LITTLE werd bewezen.

14.7.3. verblijven en wachten

Er is een heel eenvoudig verband tussen het gemiddeld aantal aanwezigen en het gemiddeld aantal wachtenden.

Een klant in het wachttijdensysteem wordt door de server(s) verwerkt of wacht. Daarom is

$$\textit{gem. aantal aanwezigen} = \textit{gem. aantal wachtenden} + \textit{gem. aantal in verwerking}$$

Hierin staat helemaal rechts het gemiddeld aantal klanten dat op een willekeurig moment bij de server(s) in verwerking is. Voor een systeem met één server is dat eenvoudig te berekenen. In zo'n systeem is het aantal klanten in verwerking 1 als de server werk heeft en 0 als de server zonder werk is. De kans dat er op een lukraak moment een klant bij de server in verwerking is, is een tijdgemiddelde kans, dus een kans op aantreffen; het is de bezettingsgraad van de server. Het gemiddeld aantal 'in verwerking' is dus voor elk systeem met één server

$$\textit{gem. aantal in verwerking} = U \times 1 + (1-U) \times 0 = U$$

Invullen levert

$$\textit{gem. aantal aanwezigen} = \textit{gem. aantal wachtenden} + U$$

Het verschil tussen het aantal aanwezigen en het aantal wachtenden is in een wachttijdensysteem met één server niet altijd 1, het is slechts 1 als er een klant in het systeem is, vandaar de *U*.

De basisrelatie tussen doorstroom, bezettingsgraad en gemiddelde bedienduur is eigenlijk niets anders is dan een geval van de relatie van Little (zie al §4.6.2). Voor een wachttijdensysteem met één server is *U* het gemiddeld aantal klanten dat bij de server in verwerking is. De doorstroom is de 'stroom' en de gemiddelde bedienduur is de gemiddelde duur van de toestand 'in verwerking bij de server'. De relatie van Little gaat met deze waarden voor gemiddeld aantal, duur en stroom over in de uitspraak dat de doorstroom gelijk is aan de bezettingsgraad, gedeeld door de gemiddelde bedienduur. Door 'Little' toe te passen op de groepen 'aanwezigen', 'wachtenden' en 'in verwerking' ontstaan dus de relaties van Little uit §14.7.2 en de basisrelatie uit §14.5. Van belang is dat in de basisrelatie de gemiddelde bedienduur optreedt, dus de gemiddelde tussenpoos bij uittreden en niet de gemiddelde verwerkingsduur of de gemiddelde serviceduur.

We kunnen de betrekking tussen het gemiddeld aantal aanwezigen en het gemiddeld aantal wachtenden verder uitwerken door voor beide de relatie van Little toe te passen. Delen van linker- en rechterlid door de doorstroom geeft, samen met de basisrelatie tussen doorstroom, bezettingsgraad en gemiddelde bedienduur, voor een 1-server systeem

$$E(\textit{verblijfduur}) = E(\textit{wachtduur}) + E(T_u)$$

of in operationele vorm

$$'\textit{verblijfduur} = '\textit{wachtduur} + '\textit{bedienduur}$$

Een relatie die misschien niet verrast. Maar merk op dat in het rechterlid de gemiddelde bedienduur staat.

Grootheden als de gemiddelde wachtduur en het gemiddeld aantal wachtenden zijn alleen relevant als er een duidelijke scheidslijn valt te trekken tussen wachten en verwerkt worden. Dit is vaak niet het geval. De opbouw van de verwerking door het samen delen van middelen maakt nogal eens dat het verschil tussen wachten en voortgang maken vervaagt. Bovendien maakt de scheduling op zich het soms onmogelijk te zeggen wanneer er gewacht wordt, dit doet zich bijvoorbeeld bij PROCESSOR SHARING voor (§14.10). De gemiddelde wachtduur krijgt in zulke gevallen de louter formele betekenis van verschil tussen gemiddelde verblijfduur en gemiddelde bedienduur.

14.7.4. samenvatting

De nu verkregen betrekkingen tussen de gemiddelde verblijfduur, de gemiddelde wachtduur, het gemiddeld aantal aanwezigen en het gemiddeld aantal wachtenden laten zien dat het voldoende is om één van deze vier grootheden te bepalen. De anderen zijn daarna eenvoudig aan te geven, mits de gemiddelde bedienduur bekend is. De betrekkingen zijn natuurlijk consistent, onderlinge manipulaties leveren geen tegenspraak en geen nieuws op. Het hangt van de omstandigheden af welke van de vier grootheden het gemakkelijkst is op te sporen. Soms is dat de gemiddelde wachtduur (bijvoorbeeld in §15.3), soms het gemiddeld aantal aanwezigen (bijvoorbeeld in §16.2), enzovoort. Is één grootheid gevonden, dan volgen de andere drie. De gevonden relaties zijn:

gemiddeld aantal aanwezigen	=	*doorstroom* × *gemiddelde verblijfduur*
gemiddeld aantal wachtenden	=	*doorstroom* × *gemiddelde wachtduur*
gemiddeld aantal aanwezigen	=	*gemiddeld aantal wachtenden* + *gemiddeld aantal in verwerking*

Voor een systeem met één server (of wanneer de servers samen als een aggregate-server worden gezien)

doorstroom	=	*bezettingsgraad/(gem. bedienduur)*
gem. aantal aanwezigen	=	*gem. aantal wachtenden* + *bezettingsgraad*
gem. verblijfduur	=	*gem. wachtduur* + *gem. bedienduur*

14.7.5. voorbeeld: disks aan een string

Twee disks zijn via eenzelfde verbinding (string) verbonden met een controller. Disk1 is 30% van de tijd bezet, disk2 15%. De doorstroom aan diskaccessen op disk1 is 50 accessen per sekonde en op disk2 30 accessen per sekonde. Door een systeem-monitor wordt gemeld dat er gemiddeld (op een lukraak moment) 0.09 diskaccessen voor disk1 wachten en 0.06 voor disk2. Wat is de verhouding van de prestatiegrootheden voor de disks? De gegevens en de resultaten staan in tabel 14.1 en tabel 14.2.

	disks		
	doorstroom	bezettings-graad	gem. aantal wachtenden
disk1	50	0.30	0.09
disk2	30	0.15	0.06
verhouding	1.66	2	1.5

Tabel 14.1. Gegevens funktioneren disks. Doorstroom per sekonde.

De gemiddelde bedienduur per acces bij disk1 volgt met de basisrelatie uit

$$50 \text{ per sek.} = 0.05 \text{ per msek.} = 0.30 / (\textit{gem. bedienduur}_1)$$

De gemiddelde wachtduur van een acces voor disk1 is volgens Little

$$\textit{gem. wachtduur}_1 = 0.09 / 0.05 = 1.8 \text{ msek.}$$

De gemiddelde verblijfduur van een acces voor disk1 is

$$\textit{gem. verblijfduur}_1 = 1.8 + 6 = 7.8 \text{ msek.}$$

Het gemiddeld aantal aanwezigen (de queuelengte) voor disk1 is

$$\textit{gem. aantal aanwezigen} = 0.09 + 0.30 = 7.8 \times 0.050 = 0.39$$

	disks			
	gem. bedienduur	gem. wachtduur	gem. verblijfduur	gem. aantal aanwezigen
disk1	6	1.8	7.8	0.39
disk2	5	2	7	0.21
verhouding	1.2	0.9	1.11	1.9

Tabel 14.2. Prestatiegrootheden disks. Samenhang met meetgegevens via Little. Tijden in msek.

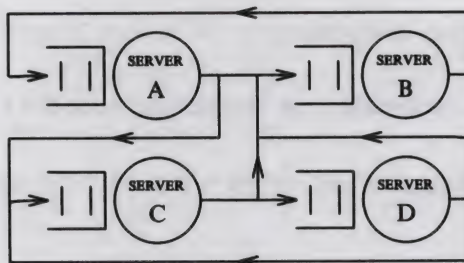
14.8. OPEN EN GESLOTEN (INTERMEZZO QUEUING-NETWERK)

Combinaties van wachttijdensystemen vallen in twee grote groepen uiteen: ze zijn *open* of *gesloten*.

In *open* systemen komen klanten van buiten het systeem binnen, worden verwerkt door de servers uit het systeem en verlaten tenslotte het systeem weer. Het gewone wachttijdensysteem rond een loket is een open systeem; ook in figuur 14.3 staat een open systeem.

In *gesloten* systemen ligt het aantal klanten vast (figuur 14.7). Deze pendelen voortdurend heen en weer tussen de servers. Er is geen stroom van klanten van buiten naar binnen. Als een individuele klant na enige tijd volledig is afgewerkt wordt hij vervangen door een andere klant, waarvan dan de verwerking begint. Daardoor verandert er niets, omdat de individualiteit van de klanten niet van belang is.

De interaktieve verwerking van opdrachten van terminalisten is een bekend voorbeeld van een gesloten systeem. Het is in het hoofdstuk RESPONSETIJD EN GEBRUIKERS en bij het voorbeeld 'accessen per usercommando' uit §9.9 (figuur 9.4) besproken. De opdrachten doen de CPU's, de randapparaten en de gebruikers aan. Tussen deze servers pendelen ze heen en weer in een queuing-netwerk.



Figuur 14.7. Gesloten queuing-netwerk, combinatie van de wachttijdensystemen bij de servers A, B, C en D. Klanten gaan van A naar B en C, van B naar A, van C naar B en D en van D naar B en C. Er komen geen klanten van buiten af binnen.

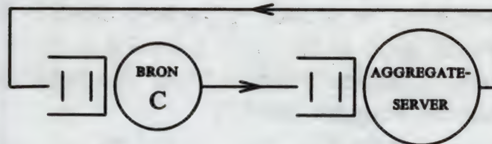
Een bezoek aan de server 'gebruikers' duurt de denktijd; er wordt daarbij afgestreept dat er een opdracht is afgewerkt, maar aansluitend wordt een vervanger ingebracht, zodat het aantal opdrachten niet verandert.

Het verschil tussen een open en een gesloten systeem wordt ook geformuleerd met het begrip *klantenbron*, een klantenbron levert klanten aan.

In een open systeem is de klantenbron onbeperkt en daarmee oneindig werkzaam: hij droogt nooit uit.

Bij een beperkte of *eindige* klantenbron is er slechts een beperkt aantal potentiële klanten; men zegt dat de *populatie* vast ligt. Klanten die het systeem verlaten worden enige tijd later opnieuw door de klantenbron ingebracht. Er vinden geen aankomsten plaats als alle potentiële klanten zich in het wachttijdensysteem bevinden, de klantenbron is dan tijdelijk uitgedroogd.

Elk gesloten systeem gaat over in een systeem met een eindige klantenbron door één van de servers als bron aan te wijzen en de andere servers samen als een 'samengestelde' server (aggregate-server) op te vatten (figuur 14.7 en 14.8). In een gesloten systeem is een klantenbron dus beperkt.



Figuur 14.8. Het gesloten queuing-netwerk uit de vorige figuur (14.7) is gelijkwaardig met systemen met een eindige bron. Hier wordt server C gezien als een bron van klanten voor een aggregate-server, die bestaat uit de servers A, B en D.

Een systeem met aankomsten volgens een Poissonproces is een open systeem, met het Poissonproces als de onbeperkte bron van klanten. Een Poissonproces gedraagt zich onder alle omstandigheden op dezelfde manier, hoe de toestand in het wachttijdensysteem ook is. Altijd komen klanten op lukrake momenten binnen.

In het gesloten systeem dat de interaktieve verwerking van opdrachten van terminalisten beschrijft, worden de gebruikers vaak als een klantenbron gezien en de computerconfiguratie als een aggregate-server die deze klanten helpt. Wanneer er bijvoorbeeld tien gebruikers bezig zijn, pendelen er in het gesloten queuing-netwerk met als servers de gebruikers, de CPU's en de randapparaten, tien opdrachten heen en weer. Er zijn dan potentieel tien klanten voor de aggregate-server 'computerconfiguratie'. Als alle tien de gebruikers in de responsefase zijn, worden er geen nieuwe opdrachten ingebracht want er zijn geen denkende terminalisten. In die situatie is de klantenbron 'gebruikers' tijdelijk uitgedroogd en zijn de tien

opdrachten bij de aggregate-server 'computerconfiguratie'. De klantenbron werkt het snelst als alle tien de terminalisten denken, er worden dan tien opdrachten per gemiddelde denktijd ingebracht.

Door een gesloten systeem op te vatten als een systeem met een beperkte klantenbron blijkt duidelijk dat bij elke server uit een gesloten systeem het aankomstpatroon reageert op de verwerking. Als er betrekkelijk veel klanten in verwerking zijn of op verwerking wachten, komen er betrekkelijk weinig per tijdseenheid binnen. Door deze terugkoppeling keert 'de wal het schip'. Daardoor zal een rij wachtenden bij een gesloten systeem nooit zo lang worden als in een open systeem bij dezelfde waarden voor de bezettingsgraad van de servers.

In het volgende hoofdstuk beperken we ons tot de bekendste open systemen. In zulke systemen blijken bij hoge bezettingsgraden van de servers erg lange wachtrijen voor te komen. Dit verschijnsel kan zich in gesloten systemen niet in die mate voordoen.

In open systemen is de doorstroom door het systeem een extern gegeven. Er is een bepaalde instroom van klanten van buiten af en die doorstroom wordt in de stationaire fase verwerkt. Het is niet druk als het systeem een betrekkelijk lage doorstroom krijgt te verwerken en als de doorstroom betrekkelijk hoog is, is het druk; in beide gevallen heeft de stroom opdrachten dezelfde samenstelling (zelfde verdeling van de serviceduur). In de open systemen is de doorstroom een maat voor de drukte, in zulke systemen betekent 'meer van hetzelfde' dat de instroom, en daarmee de doorstroom, vergroot wordt.

Het verwerken van de opdrachten van gebruikers om hun uitvoer naar een laserprinter te sturen kan gezien worden als een open systeem. De server is de laserprinter(driver), die een stroom van 'drukaf'-opdrachten te verwerken krijgt. De aanvoer van deze opdrachten zal waarschijnlijk niet of nauwelijks beïnvloed worden door het aantal wachtende opdrachten bij de laserprinter, er is maar een geringe terugkoppeling tussen aanvoer en verwerking. Daardoor is een modellering als een open systeem op zijn plaats. Alleen de drang van gebruikers om hun uitvoer te zien, bepaalt hoe groot de doorstroom door de laserprinter is. Dit geldt onder het voorbehoud dat de laserprinter het aan kan: de doorstroom moet kleiner zijn dan 'één per gemiddelde afdruktijd', anders ontstaat er geen stationaire situatie (§14.5).

In gesloten systemen is niet de doorstroom, maar het aantal klanten in het gesloten systeem de maat voor de drukte. In deze systemen kan voor elk aantal klanten de doorstroom berekend worden uit de serviceduren en de specificaties van de load-dependent servers (§14.9.2).

In het gesloten systeem 'terminalisten en configuratie' bijvoorbeeld is het aantal terminalisten s de maat voor de drukte (§6.2.2). In het gesloten systeem dat de verwerking van transakties bij een vaste multiprogrammeringsgraad beschrijft, is de multiprogrammeringsgraad de maat voor de drukte (figuur 14.9, de 'high level scheduler' regelt de toegang).

Heel vaak kan een bepaalde situatie op meerdere manieren als een wachttijdensysteem gemodelleerd worden. Het hangt van de specificaties en de vraagstelling af, welke manier te verkiezen is. Hoewel open en gesloten systemen wezenlijk verschillend zijn, kunnen ze gelukkig met soortgelijke theorema's en algoritmen geanalyseerd worden.

De responsetijdrelaties zijn typisch voor gesloten systemen. De eenvoudigste consequenties van het samen gebruik maken van een server in een gesloten systeem zijn in §5.5.2 en §6.2.5 besproken.

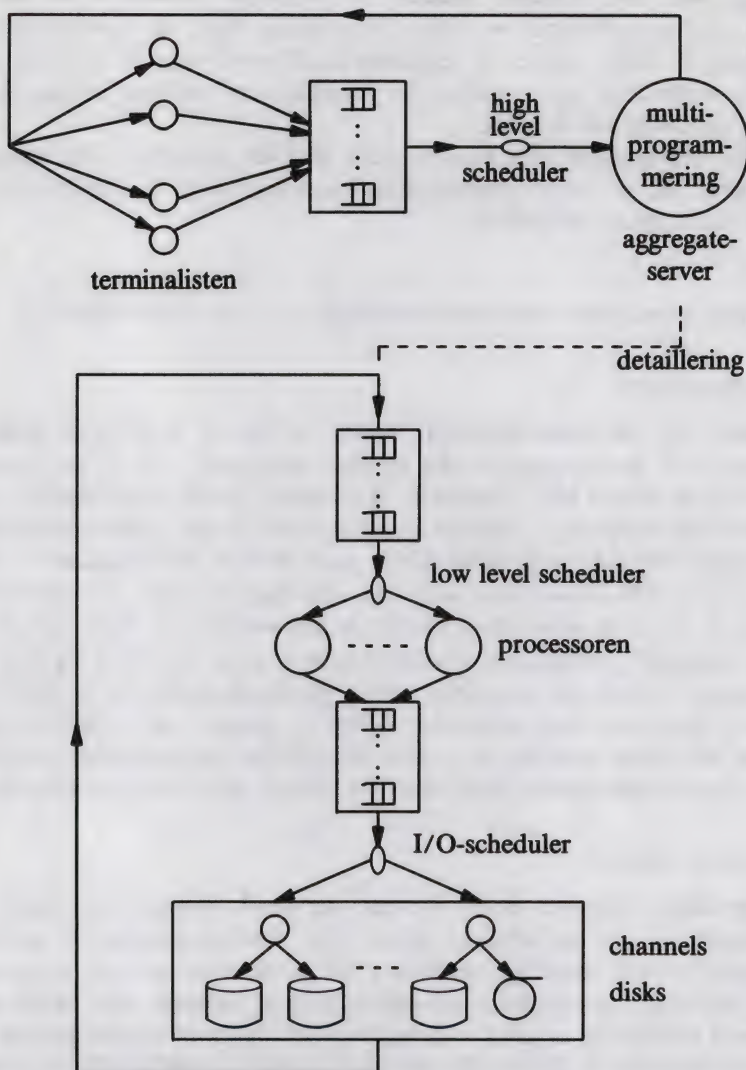
14.9. LOAD-DEPENDENT SERVERS (INTERMEZZO QUEUING-NETWERK)

14.9.1. infinite server

Er is soms bij een load-dependent server in het te analyseren gebied geen bovengrens voor de doorstroom, die hij kan verwerken. Dit is het geval als de snelheid met de drukte blijft toenemen. Een *infinite server* bijvoorbeeld werkt met een snelheid die evenredig is met het aantal klanten in zijn wachttijdensysteem. Als er 10 klanten voor hem zijn werkt hij 10 maal zo snel dan wanneer er maar één klant is, zijn er 100 klanten dan werkt hij 100 maal zo snel. De verblijfduur bij zo'n server is altijd de serviceduur, dat is de tijd waarin een klant verwerkt wordt in de als standaard gehanteerde situatie waarin er maar één klant bij hem is. Een 'infinite server' wordt ook een *delay server* genoemd omdat de afwikkeling door deze server altijd een vaste vertraging (delay) ter grootte van de serviceduur betekent. Een vertraging waarvan de grootte niet van de omstandigheden afhangt, kan binnen de wachttijdentheorie altijd door een infinite server worden gemodelleerd.

14.9.2. queuing-netwerk

In veel gevallen wordt het sneller werken van een load-dependent server veroorzaakt doordat er als het drukker wordt meer deelservers parallel werken. De laatste jaren is het mogelijk geworden dit te analyseren met eenvoudige en efficiënte algoritmen om queuing-netwerken door te rekenen. Het MVA-algoritme bijvoorbeeld is krachtig en goed te doorzien. Dit algoritme maakt gebruik van een aankomststelling (als in §15.9), die een generalisatie is van het PASTA-principe uit het hoofdstuk SIGNALEN EN POISSONPROCES. Van de servers in het systeem moeten de serviceduren bekend zijn; de servers kunnen op zich load-dependent zijn, mits hun verwerkingssnelheid alleen van het aantal klanten in hun wachttijdensysteem afhangt. De relatie van Little maakt het daarna voor gesloten netwerken mogelijk de doorstroom en diverse prestatiegrootheden iteratief uit te rekenen voor een ruime klasse van schedulingsregelingen en verdelingsfunkties van de serviceduren. Combinaties van wachttijdensystemen, die op deze manier geanalyseerd kunnen worden, heten *separabel* (§16.3), de oplossing is in 'produktvorm'. In §16.4 staan de uitkomsten voor de gemiddelde verblijfduur in open separabele queuingnetwerken.



Figuur 14.9. Computerconfiguratie, gezien als een aggregate-server. De aggregate-server is samengesteld uit processoren en randapparaten (hier disks en channels).

We zullen niet verder ingaan op het doorrekenen van queuing-netwerken. Maar in de voorgaande hoofdstukken leerden we het gesloten queuing-netwerk 'computer-configuratie' globaal kennen. Wat we daar vonden herhalen we nu in termen van wachttijdensystemen.

14.9.3. voorbeeld: computerconfiguratie

In het hoofdstuk RESPONSETIJD EN GEBRUIKERS analyseerden we de verwerking van interaktieve opdrachten door een computerconfiguratie. De configuratie kan als een aggregate-server worden opgevat met de terminalisten als de bron van opdrachten. Omdat een opdracht bij terugkeer naar de terminals steeds door zijn eigen terminalist in de denktijd verwerkt wordt, gedragen de terminalisten samen zich als een infinite server met de gemiddelde denktijd als gemiddelde serviceduur. Het geheel is een gesloten queuing-netwerk met twee servers: de infinite server 'gebruikers' en de aggregate-server 'computerconfiguratie' (figuur 14.9).

Het gedrag van de aggregate-server hebben we in §9.9 voor het voorbeeld 'accessen per usercommando' leren kennen (figuur 9.4 is een speciaal geval van figuur 14.9). Het is van belang de serviceduur per acces te onderscheiden van de serviceduur per opdracht, die in §9.9 de verwerkingsduur per opdracht (commando) wordt genoemd.

De server 'computerconfiguratie' is bij een geringe tot matige belasting sterk load-dependent. Neem bijvoorbeeld de situatie uit tabel 9.1 in §9.9.3 (de gegevens zijn overgenomen in tabel 14.3).

device	gemiddelde serviceduur per opdracht
CPU	10
paging	10
I/O-1	10
I/O-2	20
I/O-3	20
I/O-4	30

Tabel 14.3. Verwerking van de usercommando's. Gemiddelde serviceduren (verwerkingsduren, in msek.) bij de devices per opdracht (commando). De serviceduur wordt in gedeelten afgehandeld, bij elk acces wordt een deel van de verwerking afgemaakt. De devices hebben een vaste verwerkingssnelheid.

Een deel van de tijd is er één opdracht in het netwerk, samengesteld uit de servers CPU tot en met I/O-4. Er wordt door die opdracht natuurlijk niet gewacht, hij wordt in $10+10+10+20+20+30 = 100$ msek. afgewikkeld. De doorstroom is dan, volgens de relatie van Little, 10 opdrachten per sekonde. De bezettingsgraad van de aggregate-server is in die tijd 1 (want de aggregate-server is steeds met de klant bezig), zodat de gemiddelde bedienduur in de 'virtual time' waarin er één opdracht is, 100 msek. is.

Gedurende een ander deel van de tijd zijn er twee opdrachten bij de zes servers. Het is mogelijk dat een van de twee bij een van deze servers staat te wachten,

maar dat zal waarschijnlijk weinig voorkomen. Als dit wachten wordt verwaarloosd, worden de opdrachten ook in deze tijd in 100 msek. afgewikkeld en is de doorstroom, volgens Little, 20 opdrachten per sekonde. De gemiddelde bedienduur in deze tijd is 50 msek.

In de tijd dat er drie klanten zijn zal de gemiddelde bedienduur 100/3 msek. bedragen, tenminste als er ook dan niet gewacht wordt. Zolang de opdrachten elkaar in het inwendige niet in de weg zitten, zal de (effektieve) snelheid van de load-dependent server (bijna) lineair toenemen doordat er steeds meer parallel wordt gewerkt (vergelijk figuur 6.2 uit §6.2). Als het niet druk is, gedraagt de aggregate-server zich als een infinite server.

Als de drukte verder stijgt, dat wil zeggen de multiprogrammeringsgraad toeneemt, kan een van de deelservers (devices), die met een vaste snelheid werkt, 'vol' raken; zijn bezettingsgraad komt dicht bij 1. Deze server wordt *knelpunt* en gaat de snelheid van de aggregate-server, en daarmee de doorstroom, begrenzen. In tabel 14.3 is dit de fixed-rate server I/O-4 met een gemiddelde serviceduur (verwerkingsduur per opdracht) van 30 msek. De gemiddelde bedienduur bij de aggregate-server is daardoor minimaal 30 msek. De aggregate-server gaat zich als het druk wordt steeds meer als een fixed-rate server gedragen.

De servers uit het netwerk verwerken elk een verschillende doorstroom aan accessen, maar ze verwerken dezelfde doorstroom aan opdrachten. De gemiddelde serviceduur per opdracht bij de fixed-rate server I/O-2 is 20 msek. Als I/O-4 bijna volledig bezet is, is volgens de basisrelatie tussen doorstroom, bezettingsgraad en gemiddelde bedienduur de bezettingsgraad van I/O-2 minder dan $1000/30 \times 0.020 = 0.67$.

Het is mogelijk dat een van de deelservers bij een hogere multiprogrammeringsgraad trager gaat werken (meer overhead, paging en swapping per opdracht), waardoor de snelheid van de aggregate-server bij verhoging van de multiprogrammeringsgraad terugloopt (als tabel 9.3, maar met ook devices die trager worden).

De systeembeheerders zullen er steeds voor zorgen dat de multiprogrammeringsgraad niet te hoog kan stijgen (de high level scheduler regelt de toegang). De aggregate-server 'computerconfiguratie' blijft daardoor, ook bij grote drukte, als een server met een vaste snelheid werken.

Dit alles staat in andere bewoordingen reeds in §6.2.

14.10. SCHEDULING

'Scheduling' houdt een afspraak in de klanten volgens een bepaalde bedieningsregeling te helpen. Deze regeling kan bedoeld zijn om de load-dependent server sneller te maken door er voor te zorgen dat deze per klant gemiddeld minder werk hoeft te doen. Maar bij een server met een vaste verwerkingssnelheid is het doel slechts om, afhankelijk van de omstandigheden, de ene groep klanten te bevoordelen ten koste van de andere; dat moet het effekt hebben dat een bepaald

prestatie-criterium zoals de gemiddelde wachtduur zo kort mogelijk wordt. Veel gebruikte regelingen zijn

- RANDOM: de volgende klant wordt lukraak gekozen uit de wachtenden.
- FCFS (FIRST COME FIRST SERVED), heet ook FIFO: behandeling in volgorde van binnenkomst.
- LCFS (LAST COME FIRST SERVED): behandeling in omgekeerde volgorde van binnenkomst.
- RR (ROUND ROBIN): de server houdt zich nooit langer dan een afgesproken tijd (time slice) achterelkaar met eenzelfde klant bezig. Een gedeeltelijk afgewerkte klant gaat terug naar de rij wachtende klanten. Nieuw binnengekomen en onafgewerkte klanten komen in volgorde van binnenkomst in de wachtrij (FCFS) aan de beurt. De keuze van de volgende klant kan echter ook in plaats van via FCFS volgens een prioriteitsregeling verlopen.
- SJN (SHORTEST JOB NEXT), heet ook SPT (SHORTEST PROCESSING TIME FIRST): de kortste job komt het eerst aan bod. Variaties op SJN zijn SEPT en SRPT:
- SEPT (SHORTEST EXPECTED PROCESSING TIME FIRST), SRPT (SHORTEST REMAINING PROCESSING TIME FIRST).
- PS (PROCESSOR SHARING), limietgeval voor RR: de time slice is verwaarloosbaar kort (infinitesimaal). Iedere aanwezige klant krijgt continu hulp van de server, maar op een lagere verwerkingssnelheid dan wanneer hij de enige klant zou zijn. Deze individuele verwerkingssnelheid is omgekeerd evenredig met het aantal aanwezige klanten. PS is vaak een goede benadering van ROUND ROBIN met een korte time slice.
- PRIORITEITSGEWIJS (vele variaties mogelijk, bijvoorbeeld met prioriteiten die dynamisch veranderen).

De voorschriften kunnen volgens diverse principes ingedeeld worden, bijvoorbeeld naar de *hoeveelheid informatie* die de server nodig heeft om de volgende klant te kiezen: moet hij weten hoelang de klanten verwerkt gaan worden, zoals in S(HORTEST) J(OB) N(EXT), of hoeft hij alleen maar een schatting daarvan te kennen, zoals in S(HORTEST) E(XPECTED) P(ROCESSING T(IME) FIRST, of alleen het moment van aankomst, zoals bij FIFO of helemaal niets, zoals in RANDOM.

Een bedieningsregeling maakt vaak gebruik van een indeling van de klanten naar groepen. Vaak wordt aan een klantengroep een prioriteit gekoppeld. De groepsindeling en de prioriteit kunnen afhankelijk zijn van de tijd, bijvoorbeeld van de tijd die in het systeem is doorgebracht.

Belangrijk is of de regeling *preemptive* of *non-preemptive* is. Als de regeling 'preemptive' is kan de verwerking van een klant onderbroken worden om de server in staat te stellen voor een andere klant te gaan werken. Later wordt de verwerking weer voortgezet op het punt van onderbreking (het is preciezer gezegd

preemptive resume). Het voortijdig afbreken brengt altijd overhead mee, omdat vaker op een andere klant moet worden overgestapt. Door het afbreken zijn altijd extra voorzieningen nodig om de overhead klein te houden. De ROUND ROBIN scheduling is zwaar preemptive.

Elke scheduling kan formeel beschreven worden in termen van *beslissingsmomenten* en *voorkeursfunkties*. Op de beslissingsmomenten wordt beslist met welke klant verder wordt gegaan. Als de regeling 'non-preemptive' is, zijn dat de momenten waarop een klant het systeem verlaat. Wanneer de regeling 'preemptive' is, zijn ook de momenten waarop een klant binnentreedt beslissingsmomenten. Bij scheduling volgens ROUND ROBIN is het einde van een time slice een beslissingsmoment.

De voorkeursfunktie van de server maakt uit welke klant op een beslissingsmoment in verwerking wordt genomen. Argumenten van de voorkeursfunktie kunnen zijn: de serviceduur van de klant, zijn moment van aankomst, enzovoort.

14.11. NOTATIE VAN KENDALL

In open systemen is bekend hoeveel opdrachten er per tijdseenheid binnenkomen, deze doorstroom wordt verwerkt. Een bekende notatie voor de open systemen is afkomstig van de statisticus D.G. KENDALL. Er wordt steeds verondersteld dat een aankomende klant een serviceduur vraagt, die getrokken is uit de een of andere verdeling: er zal geen patroon zitten in de volgorde van aanvragen van serviceduren. Ook wordt aangenomen dat de servers altijd even snel werken: het zijn fixed-rate servers.

In de notatie van Kendall worden het aankomstproces, de verdeling van de serviceduren en het aantal servers symbolisch vastgelegd. Deze gegevens worden in deze volgorde opgeschreven, met de '/' als scheidingsteken. Als identificatie voor zowel het Poissonproces als de negatief exponentiële verdeling wordt de M genomen van Markov (A.A. MARKOV, 1856-1922, waarnaar ook de Markovketens zijn vernoemd). Als elke verdeling voor de tussenpozen of de serviceduren toegestaan is, wordt dit aangegeven met de G van *General*. De D staat voor *Deterministic*, voor vaste tussenpozen of vaste serviceduren. Verder staat E_K voor een *Erlangverdeling*. De serviceduur is bij zo'n verdeling opgebouwd uit K negatief exponentieel verdeelde fasen.

Volgens de Kendall-notatie is een M/M/1 systeem een wachttijdensysteem met aankomsten volgens een Poissonproces, negatief exponentieel verdeelde serviceduren en één server. En M/G/1 is net zo'n systeem, waarin de verdeling van de serviceduren echter algemeen is. Het is een wachttijdensysteem met aankomsten volgens een Poissonproces, de verdeling van de serviceduren kan iedere vorm hebben (binnen zekere grenzen —volgens de zuivere wiskunde—) en er is één server. In een G/G/2 systeem zijn zowel de tussenpozen bij aankomst als de serviceduren algemeen verdeeld, terwijl er twee servers de klanten helpen.

In de notatie van Kendall zijn maar enkele karakteristieken van het systeem aangegeven. De notatie zegt bijvoorbeeld niets over de scheduling, deze moet altijd apart opgegeven worden. Helaas wordt soms gedaan alsof dit wel het geval is en

neemt men aan dat in elk M/G/1 of M/M/1 systeem klanten volgens FCFS geholpen worden. Het is echter niet terecht de specificatie van de scheduling weg te laten. In een M/M/1 systeem bijvoorbeeld kan de afwikkeling volgens ROUND ROBIN geregeld zijn.

In het volgende hoofdstuk richten we ons op de open systemen met één server; in veel praktijksituaties zijn deze een goede 'eerste benadering'. We bespreken het M/G/1 en het M/M/1 systeem, dit zijn de wereldwijd vermaarde 'klassieke wachttijdensystemen'.

14.12. SAMENVATTING

Wachttijdensystemen verschillen vaak sterk van elkaar. Doorstroom, bezettingsgraad en verdeling van de serviceduur karakteriseren het funktioneren van wachttijdensystemen, ze beschrijven drukte en werklast. De gemiddelde verblijfduren, wachtduren en de gemiddelde aantallen aanwezigen en wachtenden geven globaal het presteren van het systeem weer. De basisrelatie tussen doorstroom, bezettingsgraad en gemiddelde bedienduur en de relatie van Little brengen deze grootheden met elkaar in verband.

In de klassieke wachttijdensystemen treden alleen maar servers op die met een vaste snelheid werken. Veel devices zijn echter geen fixed-rate servers, maar load-dependent servers. De (effektieve) snelheid van een server wordt aangegeven door de inverse van de gemiddelde bedienduur.

Het gedrag van open en gesloten queuing-netwerken is bij grote drukte verschillend. De klassieke wachttijdensystemen zijn open systemen, de responsetijdrelaties zijn karakteristiek voor gesloten systemen.

De in het wachttijdensysteem gehanteerde afwikkelingsregeling ('scheduling') is naast de snelheid van de server bepalend voor de prestaties. De specificaties van een open systeem worden gedeeltelijk vastgelegd in de notatie van Kendall.

In dit hoofdstuk hebben we niet veel kunnen rekenen, dat gaan we in het volgende hoofdstuk doen.

14.13. OPGAVEN

opgave 14.1: operationele schatters

Geef de operationele schatters voor de gemiddelde wachtduur, de gemiddelde verblijfduur en het gemiddeld aantal wachtenden.

opgave 14.2: vaste snelheid?

Beschrijf voor de configuratie die U dagelijks gebruikt, welke servers wel en welke gewoonlijk niet met vaste snelheid werken. Geef aan hoe de gemiddelde serviceduur en de gemiddelde bedienduur gemeten zouden kunnen worden.

opgave 14.3: infinite server

Een infinite server gedraagt zich alsof er voor iedere klant altijd een standaardserver klaar staat (§14.9.1). Daarom wordt er, zo zegt men, bij een infinite server niet gewacht. Is dit in tegenspraak met de in §14.7.3 gevonden betrekking?

opgave 14.4: vervolg gebundelde Remote-write's (7.3.5; 8.5.1; 12.5.6)

Geef aan hoe bij de gebundelde Remote-write's de bezettingsgraad en de gemiddelde bedienduur berekend worden. Controleer de betrekkingen uit §14.5.

opgave 14.5: serviceduren uit bedienduren

Bepaal de serviceduren van de opdrachten die in figuur 14.6 tot (bijvoorbeeld) tijdstip 20 binnenkomen. Neem voor de afwikkelingsregeling FCFS of ROUND ROBIN met een time slice van 1 en veronderstel (bijvoorbeeld) dat de opdrachten het systeem in volgorde van binnenkomst verlaten en dat de server met vaste snelheid werkt.

opgave 14.6: formule voor knelpunten

Een computersysteem verwerkt systeem-opdrachten naast interaktieve opdrachten, die door terminalisten worden ingebracht. De huidige doorstroom aan systeem-opdrachten is *d* opdrachten per sekonde; de huidige doorstroom aan interaktieve opdrachten is even groot, ook *d* opdrachten per sekonde.

De configuratie bestaat uit het centrale processor apparaat (CPA), een randapparaat R en een randapparaat A. Voor geen van drieën hangt de verwerkingssnelheid van de omstandigheden af.

De CPA is onder de huidige omstandigheden het deel P_{CPA} van de tijd bezet met interaktieve opdrachten en het deel Q_{CPA} van de tijd met systeem-opdrachten. De bezettingsgraden van R en A zijn idem P_R, Q_R en P_A, Q_A.

Er is veel vraag naar de faciliteiten van het systeem. Na overleg wordt afgesproken dat het aantal op het systeem aktieve terminals zal mogen worden uitgebreid, terwijl de doorstroom aan systeem-opdrachten op *d* opdrachten per sekonde gehandhaafd blijft. Men vraagt zich af hoeveel de doorstroom aan interaktieve opdrachten in de toekomst zal kunnen stijgen. Uiteindelijk wordt de toename begrensd door een van de drie devices CPA, R en A, die de 'bottleneck' gaat worden.

- Leid een uitdrukking af voor de maximale relatieve toename van de doorstroom, uitgedrukt in de aangegeven grootheden (minimum van $(1-P_i-Q_i)/P_i$ voor i is CPA, R of A).
- Iemand beweert dat altijd het meest bezette device knelpunt is. Maak duidelijk waarom deze uitspraak onjuist is.
- Moet in de redenering, die het optreden van een knelpunt voorspelt, worden aangenomen dat de servers fixed-rate servers zijn?

opgave 14.7: balanceren

Een configuratie, bestaande uit een CPU, een bepaalde disk, een bepaalde drum en andere disks, drums en randapparaten, verwerkt naast andere zaken ook opdrachten van soort A en opdrachten van soort B. Zowel de opdrachten van soort A als de opdrachten van soort B doen gemiddeld één keer de disk en acht keer de drum aan. De disk en de drum worden uitsluitend door A en B benut. Er wordt aangenomen dat CPU, disk en drum met vaste snelheid werken. Voor beide soorten opdrachten is de gemiddelde serviceduur per acces bij de disk 1/18 sekonde en bij de drum 1/32 sekonde. Ze worden gemiddeld gedurende 3/20 sekonden door de CPU verwerkt, de CPU is 5% van de tijd bezig met ander werk dan van soort A en B.

De doorstroom door het systeem aan opdrachten van soort B is 2 opdrachten per sekonde.

- De doorstroom aan opdrachten van soort A zal sterk gaan stijgen in de naaste toekomst. De doorstroom aan opdrachten van soort B zal echter (vrijwel) gelijk blijven. Hoe groot blijft de bezettingsgraad van soort B op de disk en de drum (1/9,1/2)? Hoe groot kan de doorstroom van soort A op grond van deze gegevens hoogstens worden (2)?

Men brengt een gedeelte van de software van A over van de drum naar de disk. Hierdoor wordt het gemiddeld aantal diskaccessen per opdracht van soort A met één vermeerderd en het aantal drumaccessen met één verminderd. Voor de opdrachten van soort B veranderen de gemiddelde aantallen accessen per opdracht niet. De gemiddelde serviceduur per acces bij de drum wordt door deze verandering 10% korter en de gemiddelde serviceduur per acces bij de disk wordt door de langere seektijd 20% langer.

- Hoe sterk zou nu de doorstroom van A kunnen toenemen (tot 2.79)?
- Wat zou de uitkomst zijn als de geringe veranderingen in de serviceduren niet worden meegenomen?

opgave 14.8: opstarten

Bij een device worden opdrachten die het device in de toestand 'idle' aantreffen in gemiddeld 40 msek. verwerkt. Opdrachten die het device bezet vinden worden in gemiddeld 20 msek. verwerkt (de serviceduur is korter omdat er geen extra overhead nodig is). De opdrachten komen op lukrake momenten.

- Welk deel van de opdrachten wordt in gemiddeld 40 msek. verwerkt $(1-U)$? Wat is de gemiddelde bedienduur, uitgedrukt in de bezettingsgraad?
- Er komen 25 opdrachten per sekonde. Wat is de bezettingsgraad van het device (2/3)?
- Is het device een load-dependent server? Geef het verloop van de gemiddelde bedienduur als funktie van de doorstroom.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

15

Het M/G/1 systeem

15.1. INLEIDING

Het belangrijkste 'open' wachttijdensysteem is een M/G/1 systeem waarin bij de volgorde van verwerken niet op de serviceduur van de klant wordt gelet.

In dit systeem kan de *gemiddelde wachtduur* berekend worden met drie basisrelaties uit de voorgaande hoofdstukken: de relatie van Little, de PASTA-eigenschap van het Poissonproces en de uitdrukking voor de restduur uit het hoofdstuk RESTDUREN EN WACHTDUREN. De uitkomst is

$$E(\textit{wachtduur}) = \frac{E(T_s^2)}{2E(T_s)} \frac{U}{1 - U}$$

Deze 'formule van Pollaczek-Khinchin' wordt in de praktijk vaak toegepast.

De vertraging door wachten ontstaat zowel doordat de server bij aankomst toevallig bezet is (de *wachtrestduur*), als doordat 'concurrerende' klanten eerder in verwerking worden genomen. De gemiddelde wachtrestduur wordt berekend volgens §8.3. De gemiddelde wachtduur is de gemiddelde wachtrestduur, *'opgeblazen'* door te delen door $1 - U$.

Het beroemde M/M/1 systeem is een bijzonder geval van een M/G/1 systeem. In het M/M/1 systeem is de verhouding van de gemiddelde verblijfduur (responsetijd) en de gemiddelde serviceduur gelijk aan de *'opblaasfactor'* $1/(1-U)$.

De server in een M/G/1 systeem werkt steeds met dezelfde snelheid. De prestaties worden daarom afgemeten aan de wachtduur.

15.2. SPECIFICATIE

Een M/G/1 systeem is volgens de nomenclatuur van Kendall een wachttijdensysteem met één server (§14.11).

Klanten komen volgens een Poissonproces binnen (intreeproces: *M*). Ze komen dus op lukrake momenten en blijven binnenkomen, zonder er op de een of andere manier rekening mee te houden hoeveel wachtenden er al zijn.

De klanten brengen serviceduren in, waarvan de verdeling van allerlei vorm kan zijn (verdeling serviceduren: *G*(*eneral*)). Als een klant binnenkomt op een moment dat de server zonder werk is, begint deze direkt aan de opdracht van de klant. Klanten verlaten het systeem pas als ze afgewerkt zijn. Er is altijd wachtruimte voor wachtende klanten.

De server werkt altijd even snel, wat de omstandigheden ook zijn. De werklast wordt gekarakteriseerd door de verdeling van de serviceduren. De server is ook de standaardserver en de serviceduur is de tijd die de server voor de opdracht gebruikt (§14.4.6).

De gemiddelde wachtduur in een M/G/1 systeem zal worden bepaald voor een volgorde van afwikkelen, waarin bij het selecteren van de volgende klant niet op de serviceduur wordt gelet. Of de uitverkoren klant een lange of een korte verwerkingstijd met zich mee brengt maakt niet uit; er wordt niet gediscrimineerd naar serviceduur. Een scheduling van dit type zullen we een 'P-K' regeling noemen, omdat voor zo'n regeling de formule van Pollaczek-Khinchin geldt.

Bij P-K regelingen komt de serviceduur niet in de voorkeursfunktie voor en de waarden van deze funktie zijn niet gecorreleerd met de serviceduren van de klanten, noch direkt noch indirekt; de scheduling is niet 'vertekend door serviceduur'. Als bijvoorbeeld groene klanten voorrang krijgen boven rode, moeten beide soorten klanten dezelfde verdeling van de serviceduur hebben, anders zou er indirekt een samenhang ontstaan. Bij een P-K regeling zullen wachtduur en serviceduur onafhankelijk zijn.

Het M/G/1 systeem wordt onderzocht voor P-K regelingen, die niet preemptive zijn. De bekendste regeling is FCFS. Een scheduling volgens RANDOM of LCFS is eveneens van het P-K type. In een gedistribueerd systeem gaat het schedulen meestal van de server uit, deze loopt bijvoorbeeld cyclisch de locaties af, zoals bij 'polling' of in een token-ringnetwerk. Ook deze regelingen vallen onder P-K.

In de praktijk zijn veel regelingen globaal genomen P-K regelingen. Een regeling wijkt pas duidelijk af als klanten op hun serviceduur geselekteerd worden, zoals in SHORTEST JOB NEXT of in een prioriteitsregeling waarin de groep klanten met gemiddeld de kortste serviceduur de hoogste prioriteit heeft.

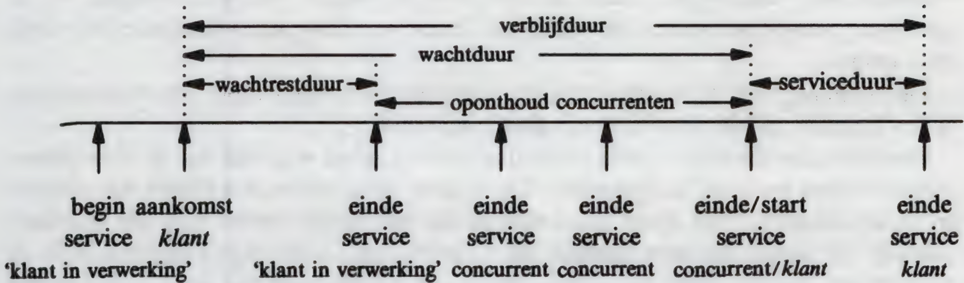
Omdat FCFS zo veel voorkomt, gaat men er vaak van uit dat bij P-K regelingen klanten volgens FCFS geholpen worden. Dat is echter niet nodig, de klasse van de P-K regelingen is veel ruimer.

In het kort komen de specificaties hier op neer: een server verwerkt klanten; de klanten komen op lukrake momenten binnen; in de volgorde waarin ze verwerkt worden speelt de serviceduur geen rol.

15.3. FORMULE GEMIDDELDE WACHTDUUR

15.3.1. aanpak

De basisoverwegingen om de gemiddelde wachtduur te vinden zijn:



Figuur 15.1. Wachtduur en verblijfduur van een *klant*. Afwikkeling non-preemptive. Op het moment van aankomst is er een klant in verwerking bij de server. Er gaan daarna drie 'concurrenten' voor.

- De toestand van het wachttijdensysteem bij aankomst is de toestand, die zich op een willekeurig moment voordoet (aankomststelling).
- de binnenkomende klant zal soms moeten wachten totdat de server de klant waarmee hij bezig is, heeft afgewerkt. Deze tijd noemen we de *wachtrestduur*.
- de binnengekomen klant zal daarna soms nog moeten wachten totdat andere klanten, die voor hem aan de beurt blijken te komen, zijn afgewerkt. Dit is oponthoud door 'concurrenten'.

15.3.2. uitwerking

-1- aankomststelling:

Bij aankomsten volgens een Poissonproces is een binnenkomende klant gelijkwaardig met een 'lukrake waarnemer' (§12.4). Deze treft een bepaalde toestand aan volgens de kans op aantreffen. Elke klant treft gemiddeld in het wachttijdensysteem het tijdgemiddelde aantal klanten aan, dus het gemiddeld aantal aanwezigen volgens §14.7.2 en hij vindt de server bezet met een kans U. Deze eigenschap van 'Poisson-aankomsten' wordt wel de 'aankomststelling voor open systemen' genoemd, het is het PASTA-principe uit §12.4.

-2- wachtrestduur:

De afwikkelingsregeling is van het P-K type en non-preemptive. Bij een non-preemptive scheduling ligt er (in virtual time) tussen het moment van uittreden van een klant en het moment van uittreden van zijn voorganger precies de serviceduur van de klant (figuur 15.1); in die tijd werkt de server met vaste snelheid voor de klant en doet er niets tussendoor. Bij een regeling van het P-K type speelt de serviceduur geen rol bij de keus van de volgende klant en er zit geen patroon in de volgorde van aanvragen, zodat er niet een of ander patroon ontstaat in de opvolging van serviceduren, bijvoorbeeld meer van lang naar kort dan van kort naar lang.

De verdeling van de tussenpozen bij uittreden (in virtual time), de bedienduren, is dus dezelfde als de verdeling van de serviceduren.

Een klant die de server bezet vindt moet in elk geval wachten tot de klant waarmee de server bezig is, is afgewerkt. De situatie is als in het hoofdstuk RESTDUREN EN WACHTDUREN. Het duurt na aankomst bij een bezette server nog een restduur, voordat de server op een andere klant overstapt. De toestandsduren zijn de tussenpozen bij uittreden (in virtual time), we stellen vast dat hun verdeling die van de serviceduren is.

De gemiddelde restduur is in §8.2 berekend, het resultaat kan in vele vormen aangegeven worden, bijvoorbeeld als (8.1)

$$E(restduur) = \frac{E(T_s^2)}{2E(T_s)} \qquad (8.1)$$

Er is een kans U dat de server bezet wordt gevonden, zodat het gemiddeld

$$E(wachtrestduur) = U\ E(restduur) \qquad (15.1)$$

duurt voordat de server na binnenkomst van de klant vrij is.

Dit is een speciaal geval van de situatie uit het hoofdstuk RESTDUREN EN WACHTDUREN. Er is steeds een kans $1-U$ dat de klant onmiddellijk in bewerking wordt genomen, er wordt dan een toestandsduur 0 aangetroffen. De uitdrukking (15.1) voor de gemiddelde wachtrestduur heeft de vorm (8.4) uit §8.3. De verschillende manieren om de gemiddelde wachtrestduur uit te rekenen zijn in §8.2-3 te vinden.

-3- concurrentie:

Vanaf het moment dat de server van klant wisselt moet in het algemeen nog verder gewacht worden en wel als de nieuwe klant 'concurrentie' ondervindt van collega's die ook geholpen willen worden. Deze bijdrage hebben we niet eerder berekend. We gaan dit nu doen met de relatie van Little.

Laten we, om de gedachten te bepalen, ons eerst indenken dat de klanten door de server volgens FCFS geholpen worden. De binnenkomende klant moet dan alle

wachtende klanten voor laten gaan. Het aantal wachtende klanten dat de binnenkomende klant gemiddeld aantreft, is het gemiddelde aantal wachtenden in het wachttijdensysteem, want het moment van aankomst van een nieuwe klant is een willekeurig moment (aankomststelling). De wachtende klanten worden gedurende hun volle serviceduur geholpen, die gemiddeld $E(T_s)$ t.e. lang is. De gemiddelde duur van het oponthoud door wachtende klanten is dus het produkt van het gemiddeld aantal wachtenden en de gemiddelde serviceduur.

Als de server de klanten volgens een P-K regeling afwerkt, geldt hetzelfde. Er zullen soms klanten die later binnenkomen, eerder geholpen worden. Klanten worden af en toe toevallig snel uitverkoren en soms moeten ze lang wachten doordat ze geen geluk hebben. Maar omdat wachtduur en serviceduur net als bij FCFS onafhankelijk zijn, zal het oponthoud dat concurrerende klanten veroorzaken, gemiddeld gelijk zijn aan

$$\textit{gem. aantal wachtenden} \times E(T_s)$$

De relatie van Little verbindt de gemiddelde wachtduur en het gemiddeld aantal wachtenden (§14.7.2). De bijdrage van de concurrenten is dus te schrijven als

$$\textit{doorstroom} \times E(\textit{wachtduur}) \times E(T_s)$$

Het gemiddelde oponthoud is hiermee uitgedrukt in de gezochte gemiddelde wachtduur.

Een klant kan alleen last van concurrenten hebben als hij de server bezet treft, dit overkomt het deel U van de klanten. Merk op dat het gemiddeld aantal storende concurrenten voor zulke klanten gelijk is aan het gemiddeld aantal wachtenden gedeeld door U. Net als bij de bepaling van de gemiddelde wachtrestduur moet worden gemiddeld over alle klanten. De voorgaande bijdrage is, analoog aan de uitdrukking voor de gemiddelde wachtrestduur, ook

$$U \times \textit{gem. aantal wachtenden} \,/\, U \times E(T_s)$$

15.3.3. gemiddelde wachtduur

De gemiddelde wachtduur is de som van de gemiddelde wachtrestduur en de gemiddelde duur van het oponthoud door concurrenten. Deze laatste bijdrage is op zich weer in de gemiddelde wachtduur uitgedrukt. Daardoor ontstaat een 'recurrente' relatie

$$E(\textit{wachtduur}) = E(\textit{wachtrestduur}) + \textit{doorstroom} \times E(\textit{wachtduur}) \times E(T_s)$$

waaruit de gemiddelde wachtduur volgt:

$$E(wachtduur) = \frac{E(wachtrestduur)}{1 - doorstroom \times E(T_s)}$$

De server uit het M/G/1 systeem is een fixed-rate server, dus (§14.5)

$$doorstroom = U / E(T_s)$$

De uitdrukking voor de gemiddelde wachtduur wordt hiermee

$$E(wachtduur) = \frac{E(wachtrestduur)}{1 - U} \qquad (15.2)$$

De gemiddelde wachtduur is de gemiddelde wachtrestduur, vermenigvuldigd met de '*opblaasfactor*' $1/(1-U)$.

Invullen van de uitdrukking voor $E(wachtrestduur)$ (15.1, 8.1) levert tenslotte de vermaarde formule van Pollaczek-Khinchin voor de gemiddelde wachtduur in een M/G/1 systeem met een non-preemptive P-K regeling:

$$E(wachtduur) = \frac{E(T_s^2)}{2E(T_s)} \frac{U}{1 - U} \qquad (15.3)$$

(naar F. POLLACZEK en naar A.Y. KHINCHIN, rond 1930 samen met de spreiding in de wachtduren afgeleid).

15.3.4. vormen van de relatie van Pollaczek-Khinchin

De gemiddelde verblijfduur in een M/G/1 systeem volgt uit de gemiddelde wachtduur, want in een 1-server systeem bij een server die steeds even snel werkt, is (§14.4.6, §14.7.3)

$$E(verblijfduur) = E(wachtduur) + E(T_s) \qquad (15.4)$$

en dus

$$E(verblijfduur) = \frac{E(T_s)}{1 - U} + (E(restduur) - E(T_s))\frac{U}{1 - U} \qquad (15.5)$$

De uitdrukking voor de restduur is op vele manieren te herschrijven, zie §8.2. Met

de variatiecoëfficiënt C van de serviceduren is volgens (8.3) uit §8.2

$$E(restduur) = (1+C^2)\, E(T_s)/2 \tag{8.3}$$

zodat

$$E(wachtduur) = \frac{U}{1-U}\,(1 + C^2)\, E(T_s)/2 \tag{15.6}$$

15.3.5. samenvatting

De gemiddelde wachtduur in het M/G/1 systeem hangt, zoals te verwachten viel, van de verdeling van de serviceduren af. Maar het is niet nodig deze verdeling helemaal te kennen, gemiddelde en variantie zijn voldoende.

De doorstroom is een gegeven (M/G/1 is een open systeem). Uit de doorstroom en de gemiddelde serviceduur volgt met de basisrelatie direkt de bezettingsgraad U en daarmee de 'opblaasfactor' $1/(1-U)$. De berekening van de gemiddelde wachtrestduur staat in §8.3. Het is opmerkelijk dat er niet meer nodig is om de gemiddelde wachtduur te bepalen.

Bij een bezettingsgraad van 25% ($U = 0.25$) is de opblaasfactor 4/3, de gemiddelde wachtduur is dan volgens (15.2) 4/3 maal de gemiddelde wachtrestduur. Bij een bezettingsgraad van 75% is de gemiddelde wachtduur 4 maal de gemiddelde wachtrestduur.

De verdere prestatiegrootheden volgen uit de gemiddelde wachtduur met de relatie van Little en de basisrelaties van wachttijdensystemen (§14.7.4).

15.4. VOORBEELD: SPREIDING IN SERVICEDUREN

De berekening van de gemiddelde wachtduur wordt geïllustreerd aan het voorbeeld 'spreiding in duren', waarvoor de gemiddelde restduur in §8.3 is berekend. We brengen het voorbeeld over naar de volgende context (tabel 8.1, 15.1 en 15.2):

Een processor verwerkt opdrachten afkomstig van diverse plaatsen uit een lokaal netwerk. De opdrachten komen volgens een Poissonproces binnen en de scheduling is van het P-K type. Er komen 2/55 opdrachten per msek.

De opdrachten zijn van diverse typen met elk een eigen verdeling van de serviceduur. Een zesde deel van de opdrachten is van type A, een derde van type B en de helft van type Z. De serviceduur van een opdracht van type A is in 2/3 van de gevallen precies 15 msek. en in 1/3 van de gevallen precies 30 msek. De serviceduur van een opdracht van type B is in 5/6 van de gevallen precies 20 msek. en in 1/6 van de gevallen precies 80 msek. Een opdracht van type Z duurt altijd precies 10 msek.

Een toestandsduur 'server bezet' correspondeert met een serviceduur. De verdeling van de toestandsduren is gelijk aan de verdeling van de serviceduren en inderdaad als in tabel 8.1. De gemiddelde restduur is in §8.3 berekend. Een binnenkomende

spreiding in serviceduren	
gem. serviceduur	$18\frac{1}{3}$
drukte	
doorstroom	$\frac{2}{55}$
bezettingsgraad	$\frac{2}{3}$
wachtrestduur	
gem. restduur	$16\frac{3}{22}$
gem. wachtrestduur	$10\frac{25}{33}$
prestaties	
gem. wachtduur	$32\frac{3}{11}$
gem. verblijfduur	$50\frac{20}{33}$
gem. aantal wachtenden	$1\frac{21}{121}$
gem. aantal aanwezigen	$1\frac{305}{363}$

Tabel 15.1. Gemiddelde wachtduur in voorbeeld 'spreiding in (service)duren' (tabel 8.1, 7.5, 15.2). De berekening van de gemiddelde restduur staat in tabel 8.1. Tijden in msek., doorstroom per msek.

opdracht die de server bezet vindt, moet gemiddeld $355/22 = 16\,3/22$ msek. wachten tot de server met de volgende opdracht gaat beginnen.

De doorstroom is $2/55$ per msek. en de gemiddelde serviceduur $55/3$ msek. De bezettingsgraad U is daarmee $2/55 \times 55/3 = 2/3$. Er is een kans van $2/3$ dat een binnenkomende klant de server bezet vindt. Het gemiddelde oponthoud door 'server bezet' is het produkt van deze kans en de gemiddelde restduur van $355/22$ msek.

$$E(\textit{wachtrestduur}) = 2/3 \times 355/22 = 355/33 = 10\frac{25}{33} \text{ msek.}$$

Tot zover loopt de berekening volgens het hoofdstuk RESTDUREN EN WACHTDUREN.

De relatie van Little legt volgens §15.3 een verband tussen de gemiddelde wachtduur en de gemiddelde wachtrestduur. De gemiddelde wachtduur wordt volgens (15.2) verkregen door de gemiddelde wachtrestduur met $1/(1-U) = 3$ te vermenigvuldigen:

$$E(\textit{wachtduur}) = (355/33)/(1-2/3) = 355/11 = 32\frac{3}{11} \text{ msek.}$$

De gemiddelde wachtduur is hier aanzienlijk langer dan de gemiddelde serviceduur. De gemiddelde verblijfduur is volgens (15.4) $355/11 + 55/3 = 50\frac{20}{33}$. De uitkomsten zijn samengevat in tabel 15.1.

15.4.1. wachtrestduur volgens schema

	spreiding in serviceduren					totaal (overall)
type	*A*		*B*		*Z*	
gem. serviceduur	*20*		*30*		*10*	55/3
	drukte					
kans optreden	1/6		1/3		1/2	
doorstroom	1/165		2/165		1/55	2/55
	optreden/aantreffen					
type	*Aa*	*Ab*	*Ba*	*Bb*	*Z*	
gem. serviceduur	*15*	*30*	*20*	*80*	*10*	
kans optreden	2/3	1/3	5/6	1/6	1	
doorstroom	2/495	1/495	1/99	1/495	1/55	2/55
bezettingsgraad	2/33	2/33	20/99	16/99	2/11	2/3
	wachtrestduur					
gem. restduur	15/2	15	10	40	5	
gem. wachtrestduur	5/11	10/11	200/99	640/99	10/11	355/33

Tabel 15.2. Bepaling van de gemiddelde wachtrestduur voor 'spreiding in serviceduren' (tabel 7.5, 8.1) uit bijdragen van de verschillende typen werk. Berekening via de doorstromen en bezettingsgraden van de typen. De restduren van de typen zijn bekend; hier zijn de serviceduren van de typen vast. Tijden in msek., doorstromen per msek.

De berekening van de gemiddelde wachtrestduur volgens het schema uit §8.3 kan ook op de volgende manier gepresenteerd worden (tabel 15.2, vergelijk tabel 8.1).

De doorstroom aan opdrachten is 2/55 per msek. Hiervan is 1/6 deel van type *A*. De doorstroom van type *A* is dus $1/6 \times 2/55 = 1/165$ opdrachten/msek. De opdrachten van type *A* vallen uiteen in twee (sub)typen, waarvan de restduren bekend zijn. Het type *Aa* komt dubbel zo vaak voor als het type *Ab*. De doorstroom van het type *Aa* is dus $2/3 \times 1/165 = 2/495$ opdrachten/msek.

De server is een deel van zijn tijd bezig met een opdracht van type *Aa*. Een opdracht van dit type wordt in gemiddeld 15 msek. verwerkt. Dus de bezettingsgraad U_{Aa} van de server door type *Aa* is $2/3 \times 1/165 \times 15 = 2/33$. Een binnenkomende opdracht vindt met een kans van 2/33 dat de server bezig is met een opdracht van type *Aa*. De kans op aantreffen van de toestand *bezig met Aa* wordt hier niet met de basisrelatie voor het verband tussen de kansen op optreden en

aantreffen bepaald als ($U = 2/3$) $\times 1/11$ (tabel 8.1), maar rechtstreeks uit de doorstroom van Aa en de basisrelatie tussen de doorstroom, de bezettingsgraad en de gemiddelde bedien(service)duur.

Een klant zal als hij Aa aantreft nog gemiddeld de gemiddelde restduur van 15/2 msek. moeten wachten. Het werk van type Aa levert daarmee een bijdrage aan de gemiddelde wachtrestduur van $2/33 \times 15/2 = 5/11$ msek. (tabel 8.1 geeft dit als ($U = 2/3$) $\times 1/11 \times 15/2$).

Op dezelfde manier vinden we de bijdragen van de andere typen uit de werklast. De gemiddelde wachtrestduur is de som van deze bijdragen.

De gemiddelde wachtrestduur wordt in deze versie van het schema berekend volgens (de som is over alle typen A uit de werklast)

$$E(wachtrestduur) = \sum_A U_A \; E(restduur_A)$$

Dit is een vorm van (8.4) uit §8.3.

Volgens het schema wordt de werklast gesplitst in componenten waarvan de gemiddelde restduur bekend is. Zulke componenten hebben vaste of negatief exponentieel verdeelde serviceduren, of een verdeling waarvan de variatiecoëfficiënt bekend is. Van elke component wordt de doorstroom bepaald. Uit de gemiddelde serviceduur van de component volgt de bezettingsgraad. Deze bezettingsgraad door werk van de component is een kans op aantreffen, die in §8.3 nog met de basisrelatie uit BEURTEN EN KANSEN wordt berekend. De bijdrage van een component A uit de werklast aan de gemiddelde wachtrestduur is het produkt van de bezettingsgraad U_A en de gemiddelde restduur $E(restduur_A)$ voor de tijd dat de component de server bezet houdt.

De doorstromen en de bezettingsgraden staan centraal. De opsplitsing naar type werk gebeurt op een voor een systeembeheerder natuurlijke manier (§5.6.1, §5.8). Als een monitor de doorstromen en de bezettingsgraden heeft aangegeven, kan de gemiddelde serviceduur worden uitgerekend. Van elk type moet (een benadering van) de spreiding in de serviceduur bekend zijn.

Merk op dat de berekeningen zo eenvoudig zijn doordat de opdrachten op lukrake momenten komen. Een opdracht van type Aa vindt met een kans U_{Aa} dat de server bezig is met werk van zijn type. Een opdracht van type B vindt met precies dezelfde kans dat de server bezig is met type Aa. Het maakt geen verschil of de server bezig is met het 'eigen' type. Er is geen compensatie nodig in de geest van 'de kans op een ander type moet groter zijn dan de kans op het eigen type, omdat de binnengekomen klant al van het eigen type is'.

	spreiding in serviceduren					totaal (overall)
type	*Aa*	*Ab*	*Ba*	*Bb*	*Z*	
serviceduur	15	30	20	80	10	18 1/3
doorstroom	2/495	1/495	1/99	1/495	1/55	2/55
	prestaties (gemiddelden)					
wachtduur	32 3/11	32 3/11	32 3/11	32 3/11	32 3/11	32 3/11
verblijfduur	47 3/11	62 3/11	52 3/11	112 3/11	42 3/11	50 20/33
wachtenden	142/1089	71/1089	355/1089	71/1089	639/1089	1 21/121
aanwezigen	208/1089	137/1089	575/1089	247/1089	837/1089	1 305/363

Tabel 15.3. Prestatiegrootheden 'spreiding in serviceduren'. Onder M/G/1 hebben alle typen dezelfde gemiddelde wachtduur, maar de gemiddelde verblijfduren (responsetijden) verschillen.

15.4.2. prestatiegrootheden

Ook voor elk type apart kunnen de gemiddelde verblijfduur en de gemiddelde aantallen klanten worden aangegeven. De berekeningen staan in tabel 15.3. De 'overall' gemiddelde wachtduur en verblijfduur volgen, zoals in §5.6.1, uit de gemiddelde wachtduur en verblijfduur voor de typen door te middelen met het relatieve aandeel in de doorstroom als gewicht. Deze gewichten zijn de kansen op optreden in de verdeling van de serviceduur, ze moeten ook in de berekening van $E(T_s^2)$ worden gebruikt (§7.2.4, §8.3, opgave 15.1).

In een M/G/1 systeem onder P-K lopen alle klanten gemiddeld dezelfde vertraging door wachten op. Maar dat betekent niet dat ze allemaal gemiddeld even lang in het systeem zitten.

15.4.3. voorbeeld: invloed restduren

Uit monitorgegevens blijkt dat de werklast voor een bepaald device voornamelijk uit vijf componenten is opgebouwd. Het device verwerkt per msek. 2/495 opdrachten van de eerste component, de bezettingsgraad door werk van deze component is 2/33. Van de tweede component worden 1/495 opdrachten per msek. verwerkt, de bezettingsgraad door werk van de tweede component is 2/33. De systeembeheerder schat dat de opdrachten van de eerste component weinig spreiding in de verwerkingsduur hebben en dat de opdrachten van de tweede component veel spreiding hebben met als variatiecoëfficiënt 2. Het device verwerkt de opdrachten met een vaste verwerkingssnelheid, de afwikkeling verloopt volgens een P-K regeling.

Uit de doorstromen en de bezettingsgraden volgen de serviceduren. De verdere specificaties (§8.4.2 met Z exponentieel verdeeld) en de berekening van de gemiddelde wachtduur staan in tabel 15.4. Door het verschil in spreiding ten opzichte van de situatie uit tabel 15.2 neemt de gemiddelde wachtduur met $13\frac{7}{11}$ toe.

	invloed restduren					totaal
	optreden/aantreffen					
component	*Aa*	*Ab*	*Ba*	*Bb*	*Z*	
verdeling	vast	$C=2$	vast	vast	neg.exp	
doorstroom	2/495	1/495	1/99	1/495	1/55	2/55
bezettingsgraad	2/33	2/33	20/99	16/99	2/11	2/3
gem. serviceduur	15	30	20	80	10	
	wachtrestduur					
gem. restduur	15/2	75	10	40	10	505/33 =
gem. wachtrestduur	5/11	50/11	200/99	640/99	20/11	15 10/33
	wachtduur					
gem. wachtduur	505/11 = 45 10/11					45 10/11

Tabel 15.4. **Berekening van de invloed van de gemiddelde restduren op de gemiddelde wachtduur. Gegevens als in tabel 15.2, alleen andere spreidingen. De verdeling van de serviceduur van de componenten *Aa* tot en met *Z* is respectievelijk vast, variatiecoefficient 2, vast, vast, negatief exponentieel. Berekening via doorstromen en bezettingsgraden. Tijden in msek., doorstromen per msek.**

15.4.4. samenvatting

De gemiddelde wachtduur wordt bepaald door eerst de gemiddelde wachtrestduur uit te rekenen. Hiervoor wordt gebruik gemaakt van het 'schema' uit §8.3, waarbij de doorstroom direkt de kansen op aantreffen geeft. Het is ook mogelijk de gemiddelde wachtrestduur rechtstreeks via de grootheid $E(T_s^2)$ uit te rekenen (§8.3, opgave 15.1), maar de berekeningen met het schema zijn eenvoudig en inzichtelijk en sluiten aan bij wat een monitor aangeeft.

De gemiddelde wachtduur volgt uit de gemiddelde wachtrestduur door te vermenigvuldigen met de opblaasfactor. De waarde van de opblaasfactor wordt door de bezettingsgraad vastgelegd.

15.5. VOORBEELD: I/O-OPDRACHTEN

Het is gebruikelijk bij het bespreken van de prestaties van een complex randapparaat Poisson-aankomsten te veronderstellen. Een stroom aan I/O-opdrachten, die van diverse kanten bij een server 'disk' binnenkomt, blijkt goed te beschrijven te zijn als een open bron van aankomsten, die op lukrake momenten opdrachten gegenereert.

De gemiddelde verblijfduur (responsetijd) bij een diskserver hebben we in §8.9 in een karakteristiek geval bekeken. Het wachttijdensysteem werd gemodelleerd als een M/G/1 systeem en wel als een M/M/1 systeem: de diskserver zou een fixed-rate server zijn met een negatief exponentieel verdeelde serviceduur.

	I/O-opdrachten		totaal
component	*hoofd*	*neven*	
verdeling	*C=1*	*C=0.5*	
gem. serviceduur	*23.5*	*10.5*	
	drukte		
bezettingsgraad	0.30	0.10	0.40
doorstroom	12.77	9.52	
	wachtrestduur (gem.)		
restduur	23.5	6.6	
wachtrestduur	7.05	0.66	7.71
	prestaties (gem.)		
wachtduur	12.84	12.84	
responsetijd	36.34	23.34	

Tabel 15.5. Bepaling van de gemiddelde responsetijd (verblijfduur) voor I/O-opdrachten. Verblijfduur is responsetijd genoemd. Tijden in msek., doorstromen per sek.

Het modelleren van de diskserver als een M/G/1 systeem geeft in de praktijk goede resultaten. Als er nauwkeurige informatie nodig is, zijn er meer gedetailleerde modellen beschikbaar, zie de literatuur.

De spreiding in de serviceduren wordt veroorzaakt door spreiding in de seektijd, rotational delay, search- en transfertijd en missed reconnect delay. In eerste benadering zijn deze grootheden onafhankelijk en kunnen de varianties opgeteld worden tot de variantie in de serviceduur. Doordat de rotational delay en de transfertijd weinig spreiding hebben is de variatiecoëfficiënt meestal kleiner dan 1 en vaak ongeveer 0.5. De verdeling van de serviceduren bij een disk is dus hypo-exponentieel (§7.7.2), de spreiding is geringer dan in een M/M/1 systeem. In dit opzicht is de berekening in §8.9 niet helemaal realistisch.

Als de werklast voor de disk uit diverse componenten bestaat met elk een eigen karakteristiek gedrag voor wat betreft seektijd, rotational delay, search- en transfertijd en missed reconnect delay, resulterend in een eigen gemiddelde serviceduur en variatiecoëfficiënt, kan de gemiddelde responsetijd voor iedere component berekend worden met de formule van Pollaczek-Khinchin.

Veronderstel bijvoorbeeld dat er naast de 'hoofdcomponent' uit §8.9 (met een variatiecoëfficiënt 1) ook een 'nevencomponent' is met een variatiecoëfficiënt 0.5, de disk wordt bijvoorbeeld gebruikt door twee verschillende applicaties. De doorstroom aan I/O-opdrachten van de hoofdcomponent (gemiddelde serviceduur 23.5 msek.) is 12.77 per sekonde, van de nevencomponent (gemiddelde serviceduur 10.5 msek.) 9.52 per sekonde. De berekening van de gemiddelde responsetijd (verblijfduur) is samengevat in tabel 15.5. Het overall gemiddelde voor de responsetijd volgt door middeling met als gewichten 12.77/22.29 en 9.52/22.29.

15.6. VERVOLG VOORBEELD: LIJNBERICHTEN (14.5.2)

In dit voorbeeld komen berichten niet op lukrake momenten, want na elke aankomst duurt het minstens 20 msek. voor de volgende klant komt. De klanten gedragen zich niet als een lukrake waarnemer, de opeenvolgende momenten van aankomst zijn gecorreleerd (§7.2.2, §7.6). Een serviceduur is hoogstens 18 msek., zodat elk nieuw bericht de lijn vrij zal vinden. De kans de server bezet te vinden is niet de bezettingsgraad, maar nul. Daardoor is elke wachtduur nul.

Laten we dit vergelijken met de situatie dat de berichten wel lukraak binnenkomen, dus volgens een Poissonproces.

Berichten komen gemiddeld 25 msek. na elkaar. Het duurt volgens de basiseigenschap van het Poissonproces vanaf het versturen van het eerste bit nog een negatief exponentieel verdeelde tijdsduur van gemiddeld 25 msek., totdat het volgende bericht komt. Er is een vrij grote kans dat er snel weer een bericht komt en er gewacht wordt.

De serviceduren zijn gemiddeld 15 msek. met een variatiecoëfficiënt van 0.5. De doorstroom is 1 per 25 msek. De bezettingsgraad is $15/25$. De gemiddelde restduur is $15/2 \times (1 + 1/4)$. Met aankomsten op lukrake momenten is de gemiddelde wachtduur volgens de formule van Pollaczek-Khinchin niet nul, maar (15.2)

$$E(wachtduur) = 3/5 \times 75/8 \times 5/2 = 14.1 \text{ msek.}$$

15.7. VOORBEELD: SNELLER OF MEER

In een lokaal netwerk funktioneert een servermachine als fileserver. De fileserver beheert een grote disk met de files van alle gebruikers uit het netwerk. Men vindt dat de responsetijden bij de fileserver te lang zijn. Daarom vraagt men zich af hoe sterk deze zullen dalen door een nieuwe server te installeren, die dubbel zo snel is als de huidige (alternatief 'sneller'). Er kan ook een extra server (van het aanwezige type) bijgekocht worden (alternatief 'meer'). We gaan de beide alternatieven vergelijken qua gemiddelde prestaties.

15.7.1. modellering

De servermachine heeft een betrekkelijk snelle CPU en een snelle disk. Omdat het werk dat hij moet doen sterk I/O-bound is, zal de snelheid van de aggregate-server 'servermachine' voornamelijk bepaald worden door de verwerkingssnelheid van de disk. Noch de CPU, noch het net zal knelpunt zijn.

De aggregate-server 'servermachine' wordt gemodelleerd als een M/G/1 systeem met een afwikkelingsvolgorde van het P-K type, non-preemptive. Er wordt een open systeem genomen omdat de terugkoppeling tussen aanvoer en responsetijd gering zal zijn (§14.8). Omdat de aankomsten ongecoördineerd van diverse plaatsen binnenkomen, is het volgens de overwegingen uit §12.4 redelijk een

Poissonproces voor de aankomsten te nemen. Er wordt verondersteld dat de aggregate-server met een vaste snelheid werkt voor de range van belastingen, waarin men geïnteresseerd is.

Omdat het net helemaal geen knelpunt is, zal er bij het transport over het net nauwelijks gewacht worden. Het vervoer over het net kan daarom als een infinite server (delay server) worden gemodelleerd (§14.9.1). De gemiddelde verwerkingsduur, en daarmee de gemiddelde verblijfduur bij de infinite server, voor de transfers over het net is 0.3 msek. Deze tijd kan bij de gemiddelde verblijfduur bij de server 'fileserver', die aanstonds wordt uitgerekend, worden opgeteld. Het resultaat is de (totale) gemiddelde responsetijd bij het gebruik van de fileserver.

Voor de serviceduren bij de fileserver nemen we weer de waarden uit het voorbeeld 'spreiding in serviceduren' (§15.4).

15.7.2. sneller

Wat zijn de consequenties van een versnelling van de server met een factor 2?

Alle verwerkingsduren worden gehalveerd. Daardoor worden ook alle restduren, en dus de gemiddelde restduren, gehalveerd. De doorstroom verandert niet (open systeem). De bezettingsgraad wordt dus een factor 2 kleiner.

De gemiddelde wachtrestduur daalt met een factor 4, omdat zowel de kans dat de server bezet is als de gemiddelde restduur een factor 2 dalen. Het voordeel werkt dubbel op!

De invloed van het oponthoud door concurrenten wordt beschreven door de opblaasfactor. Deze daalt van $1/(1-U)$ naar $1/(1-U/2)$. Omdat de opblaasfactor een niet-lineair verband aangeeft kan het verschil aanzienlijk zijn. Voor de bezettingsgraad van 2/3 uit §15.4 daalt de factor van 3 naar 3/2 en wordt dus ook gehalveerd.

De gemiddelde wachtduur wordt bij een bezettingsgraad van 2/3 dus gereduceerd met een factor 8. De gemiddelde verblijfduur (15.4) bestaat uit de gemiddelde serviceduur, die een factor 2 daalt en de gemiddelde wachtduur, die een factor 8 daalt. De reduktie in de gemiddelde verblijfduur (responsetijd) is dus altijd meer dan een factor 2. De daling is sterker naarmate de wachtduur een groter deel uitmaakt van de verblijfduur, vooral bij hoge bezettingsgraden is de winst aanzienlijk.

De berekeningen zijn samengevat in tabel 15.6.

15.7.3. meer

Als alternatief wordt geopperd een server van het type dat nu gebruikt wordt, er bij aan te schaffen. De investering zal dan geringer zijn. Zullen de twee servers samen dubbel zo snel zijn als de huidige server en dus ongeveer hetzelfde voordeel voor de prestaties bieden als de enkele dubbel zo snelle servermachine?

We kunnen de bestanddelen van de gemiddelde wachtduur weer apart bekijken en deze naïeve intuïtie controleren.

De helft van de gebruikers zal op de extra fileserver worden aangesloten. De beide servers zullen separaat werken, ze delen geen gemeenschappelijke wachtrij;

sneller of meer				
server	*normaal*	*normaal*	*normaal*	*snel*
	drukte			
gem. serviceduur	18 1/3	18 1/3	18 1/3	9 1/6
doorstroom	2/55	1/55	1/55	2/55
bezettingsgraad	2/3	1/3	1/3	1/3
	wachtrestduur (gemiddelden)			
restduur	355/22	355/22	355/22	355/44
wachtrestduur	355/33	355/66	355/66	355/132
	prestaties (gemiddelden)			
wachtduur	32 3/11	8 3/44	8 3/44	4 3/88
verblijfduur	50 20/33	26 53/132	26 53/132	13 53/264

Tabel 15.6. Vergelijking van de prestaties van twee onafhankelijke 'normale' servers of een enkele dubbel zo snelle 'snelle' server. De twee 'normale' servers verwerken samen de werklast uit tabel 15.2. De 'snelle' server verwerkt deze werklast alleen. Ook de uitgangssituatie met een enkele 'normale' server is opgenomen. Tijden in msek., doorstromen per msek.

het zijn afzonderlijke wachttijdensystemen. In de nieuwe situatie is de doorstroom bij elk van de twee servers dus de helft van de huidige doorstroom. De gemiddelde serviceduur is niet veranderd, zodat de bezettingsgraad van elke server de helft is van de oorspronkelijke. Daar de serviceduren en de restduren niet veranderd zijn, betekent dit een halvering van de gemiddelde wachtrestduur. De opblaasfactor wordt door de halvering van de bezettingsgraad weer gehalveerd. De gemiddelde wachtduur zakt dus bij elke server een factor 4.

De gemiddelde verblijfduur bestaat naast de gemiddelde wachtduur, die tot een kwart gereduceerd wordt, uit de gemiddelde serviceduur, die niet veranderd is. De vermindering van de gemiddelde verblijfduur zal niet spectaculair zijn, tenzij de gemiddelde wachtduur aanzienlijk is (tabel 15.6).

15.7.4. vergelijking

Uit de berekening van de gemiddelde verblijfduur blijkt dat deze bij de snelle server veel korter is dan bij de combinatie van twee normale servers. Vanuit de klant gezien is de snelle server dus aanzienlijk 'sneller'; de prestaties met de ene snelle server zijn gunstiger.

Merk op dat de beide normale servers samen, net als de ene snelle server apart, in staat zijn een doorstroom van hoogstens 6/55 opdrachten per msek. te verwerken. Bij een doorstroom lager dan dit maximum ontstaat er een stationaire toestand, waarin alle binnenkomende opdrachten na enige tijd verwerkt worden. Het verschijnsel dat sommige klanten niet verwerkt worden, heet in de theorie van

operatingsystemen ‘starvation’. Zolang de doorstroom kleiner dan 6/55 is, treedt dit bij geen van beide alternatieven op. Het duurt bij een hoge bezettingsgraad wel lang voordat de stationaire toestand is bereikt (lange transient state).

Als er alleen op de doorstroom zou worden gelet, presteren de alternatieven even goed. De maximale (verwerkings)snelheid van de ‘aggregate-server’, bestaande uit de combinatie van de twee normale servers is even groot als de snelheid van de ‘snelle’ server. Hier blijkt weer dat het begrip ‘snelheid’ steeds goed moet worden gespecificeerd, anders gedraagt het zich als een kameleon.

Welke aanschaf de voorkeur zal verdienen hangt natuurlijk ook van andere factoren dan de responsetijd af. Zo kan het onderhoud en de kans op uitvallen een rol spelen, die in de situatie met twee servers beter zijn.

Wat we hier in een bepaald speciaal geval gezien hebben, geldt algemeen. Twee langzame servers moeten het qua prestaties altijd afleggen tegen een dubbel zo snelle, omdat ze samen niet continu op de dubbele verwerkingsnelheid draaien. Als er toevallig maar één klant is, werken ze samen op de halve snelheid. En individueel zijn ze traag, wat zich uit in de gemiddelde verwerkingsduur en daarmee in de gemiddelde verblijfduur.

Het is nooit verstandig een verwerkingscapaciteit statisch te verdelen tussen een aantal gebruikers. De ene server kan dan wachtenden hebben, terwijl de andere server zonder werk is. Het is beter dynamisch te delen, dan treedt dit euvel niet op. De twee ‘normale’ servers zouden dan een enkele gemeenschappelijke wachtrij moeten hebben (zoals in figuur 14.4 uit §14.3, in dit geval zou dat ‘load sharing’ heten). De gemiddelde wachtduren blijken dan van de orde van grootte van die bij de snelle server te zijn, maar het verschil in verwerkingsduur en daarmee in verblijfduur blijft.

15.8. BUSY PERIOD IN M/G/1 SYSTEEM

In het hoofdstuk RESTDUREN EN WACHTDUREN (§8.7) vonden we een uitdrukking voor de gemiddelde duur van een *busy period*. In zo’n periode wordt er door de server achterelkaar ‘continu’ gewerkt (figuur 14.6 in §14.4.3). Continu werken is werken zonder idle te worden, er kunnen wel onderbrekingen zijn door scheduling (einde time slice, vertrek klant, enzovoort).

De gemiddelde duur van een ‘busy period’ is volgens §8.7 bij aankomsten volgens een Poissonproces $E(continu) = E(tussenpoos) \times U/(1-U)$. Herschrijven met de basisrelatie voor doorstroom, bezettingsgraad en gemiddelde bedienduur geeft voor een M/G/1 systeem

$$E(continu) = \frac{E(T_s)}{1-U}$$

In de gemiddelde busy period verwerkt de server dus gemiddeld $1/(1-U)$ opdrachten achterelkaar.

15.9. HET M/M/1 SYSTEEM

Een M/M/1 systeem is volgens de nomenclatuur van Kendall een systeem met één (fixed-rate) server, aankomsten volgens een Poissonproces en negatief exponentieel verdeelde serviceduren. Het is het bekendste wachttijdensysteem.

Een M/M/1 systeem is een M/G/1 systeem waarin de verdeling van de serviceduren als negatief exponentieel is gespecificeerd. De voorgaande berekeningen voor de prestatiegrootheden in een M/G/1 systeem gelden daarmee ook voor een M/M/1 systeem.

Een M/M/1 systeem is belangrijk eenvoudiger dan een M/G/1 systeem, omdat het uittreden door de negatief exponentiële verdeling van de serviceduren op lukrake momenten 'in virtual time' plaats vindt. Bij het M/M/1 systeem komt weer naar voren dat 'lukraak' vereenvoudigend werkt.

In een M/M/1 systeem is de gemiddelde restduur gelijk aan de gemiddelde serviceduur, want de variatiecoëfficiënt van de serviceduur is 1. Het duurt vanaf een willekeurig moment, en in het bijzonder vanaf de binnenkomst van een nieuwe klant, gemiddeld nog de gemiddelde serviceduur voordat de klant die in verwerking is, is afgewerkt; er is geen geheugen (§13.2).

De uitdrukkingen voor de gemiddelde wachtduur en de gemiddelde verblijfduur voor een M/M/1 systeem met een non-preemptive P-K afwikkelingsregeling volgen uit (15.1-6)

$$E(wachtrestduur) = U\,E(T_s)$$

$$E(wachtduur) = \frac{U}{1-U}\,E(T_s)$$

$$E(verblijfduur) = \frac{E(T_s)}{1-U} \tag{15.7}$$

De prestatiegrootheden zijn (§14.7.4):

gem. verblijfduur	=	*gem. serviceduur / (1 − bezettingsgraad)*
gem. wachtduur	=	*gem. serviceduur × bezettingsgraad / (1 − bezettingsgraad)*
gem. aantal aanwezigen	=	*bezettingsgraad / (1 − bezettingsgraad)*
gem. aantal wachtenden	=	*bezettingsgraad*^{2} */ (1 − bezettingsgraad)*

In een M/M/1 systeem heeft de gemiddelde verblijfduur een eenvoudige vorm. Het is de gemiddelde serviceduur, 'opgeblazen' met de factor $1/(1-U)$.

Als de verwerking zonder wachten gemiddeld x sekonden kost, duurt het verblijf bij een bezettingsgraad U gemiddeld niet x sekonden, maar $x/(1-U)$ sekonden. Een server, die een vijfde deel van zijn tijd vrij is, lijkt dus voor de gebruikers van een M/M/1 systeem gemiddeld vijfmaal langzamer dan een server die steeds vrij is.

Het rekenwerk bij een M/M/1 systeem wordt eenvoudig als direkt uit de doorstroom en de gemiddelde serviceduur de bezettingsgraad wordt berekend. De gemiddelde verblijfduur volgt daarna door de gemiddelde serviceduur te vermenigvuldigen met de 'opblaasfactor'. Uit de gemiddelde verblijfduur kunnen met de relatie van Little en de basisrelaties voor wachttijdensystemen de andere prestatiegrootheden worden berekend. Zo wordt het werken met formules vermeden.

De uitdrukking voor de gemiddelde verblijfduur is zo beknopt, omdat de recurrente betrekking voor de wachtduur uit §15.3.3 eenvoudig is. De gemiddelde wachtrestduur is $U\,E(T_s)$, dat is het produkt van $E(T_s)$ en het gemiddeld aantal klanten in verwerking (§14.7.3). Het oponthoud door concurrenten is het produkt van $E(T_s)$ en het gemiddeld aantal wachtende klanten. Samen levert dat

$$E(\mathit{wachtduur}) \;=\; \mathit{gem.\ aantal\ aanwezigen} \times E(T_s)$$

Dit is een voor de hand liggende 'ruwe afschatting' voor de gemiddelde wachtduur in een wachttijdensysteem met een fixed-rate server. In het M/M/1 systeem is het, dank zij het lukrake karakter, een correcte uitdrukking voor de gemiddelde wachtduur. 'Lukraak' ondersteunt eenvoudig denken! Met Little volgt uit

$$E(\mathit{verblijfduur}) = E(T_s) \times (1 + \mathit{gem.\ aantal\ aanwezigen})$$

dat

$$E(\mathit{verblijfduur}) = E(T_s) \times (1 + E(\mathit{verblijfduur}) \times \mathit{doorstroom})$$

Hier staat dat de gemiddelde verblijfduur ontstaat door 'opblazen' van de gemiddelde serviceduur.

15.9.1. voorbeeld: elitair

Een operatingsysteem geeft systeemwerk voorrang bij de CPU's. Het systeemwerk, dat afkomstig is van het I/O-systeem en van terminal-I/O, wordt via interrupts aangeboden. Een interrupt wordt direkt door de CPU afgehandeld, waarbij de bijbehorende systeemopdracht wordt opgeslagen. Wanneer de CPU met systeemwerk bezig is gaat hij na de interrupt daarmee verder, anders begint hij aan de aangeboden systeemopdracht. Het systeemwerk wordt volgens FCFS afgewikkeld.

negatief exponentieel	
gem. serviceduur	$18\frac{1}{3}$
drukte	
doorstroom	$\frac{2}{55}$
bezettingsgraad	$\frac{2}{3}$
verblijfduur	
gem. verblijfduur	55
prestaties	
gem. wachtduur	$36\frac{2}{3}$
gem. aantal wachtenden	$1\frac{1}{3}$
gem. aantal aanwezigen	2
wachtrestduur	
gem. restduur	$18\frac{1}{3}$
gem. wachtrestduur	$12\frac{2}{9}$

Tabel 15.7. Berekening van de prestaties van een M/M/1 systeem. De kortste manier is de gemiddelde serviceduur 'op te blazen' tot de gemiddelde verblijfduur. De standaard M/G/1 manier is via de gemiddelde wachtrestduur. Tijden in msek., doorstroom per msek.

Doordat het systeemwerk prioriteit krijgt en zelfs ander werk verdringt, is het voor deze elite alsof de CPU alleen oog voor hen heeft. Als de interrupts op lukrake momenten binnenkomen, is het afwikkelen van systeemwerk een M/G/1 wachttijdensysteem (we verwaarlozen de tijd ('overhead') die nodig is om een interrupt te verwerken).

We veronderstellen dat de gemiddelde CPU-tijd van een systeemopdracht 55/3 msek. is en dat deze tijden negatief exponentieel verdeeld zijn. De doorstroom aan systeemopdrachten is 2/55 per msek., zodat de bezettingsgraad door systeemwerk 2/3 is.

De gemiddelde verblijfduur (responsetijd) voor systeemopdrachten is $(55/3)/(1-2/3) = 55$ msek. Het duurt na de interrupt gemiddeld 55 msek. voordat de aangeleverde systeemopdracht verwerkt is. De gemiddelde wachtduur is $55 - 55/3 = 36.7$ msek. De berekening van de prestatiegrootheden is samengevat in tabel 15.7. Het enige verschil met de situatie in tabel 15.1 is de negatief exponentiële verdeling van de serviceduur. In de berekening staat nu niet de wachtduur, maar de verblijfduur centraal.

Het uitvoeren van het andere werk gebeurt tijdens perioden van 'leegloop' van de server uit het M/G/1 systeem voor de systeemopdrachten. De gemiddelde duur van zo'n periode is, omdat de systeemopdrachten volgens een Poissonproces binnenkomen, 55/2 msek. Deze perioden worden afgewisseld met 'busy periods' waarin alleen systeemwerk wordt verwerkt. Deze perioden zijn volgens §15.8 gemiddeld (55/3)/(1/3) = 55 msek. lang.

15.10. GEMIDDELDE WACHTDUUR EN SPREIDING IN SERVICEDUREN

De mate van spreiding van de serviceduren wordt aangegeven door de variatiecoëfficiënt C. Bij verdelingen met $C>1$, de hyper-exponentiële verdelingen, is volgens §8.2.1 de gemiddelde restduur groter dan de gemiddelde serviceduur, bij verdelingen met $C<1$, de hypo-exponentiële, is de gemiddelde restduur kleiner dan de gemiddelde serviceduur. De negatief exponentiële verdeling ligt daar tussen in. Volgens de uitdrukking voor de gemiddelde verblijfduur (§15.3.4, (15.5)) wordt er bij hyper-exponentieel verdeelde serviceduren gemiddeld langer gewacht dan in het overeenkomstige M/M/1 systeem, bij hypo-exponentieel verdeelde serviceduren gemiddeld korter.

De kortste gemiddelde wachtduur treedt op als de serviceduren geen spreiding hebben, dus vast zijn. Het systeem is dan een M/D/1 systeem. In zo'n systeem is de gemiddelde restduur de helft van de gemiddelde serviceduur en de gemiddelde wachtduur is

$$E(wachtduur) = \frac{U}{1-U} E(T_s)/2$$

In een M/D/1 systeem wordt gemiddeld half zo lang gewacht als in een M/M/1 systeem met dezelfde doorstroom en gemiddelde serviceduur.

Voor iedere andere verdeling van de serviceduren is de gemiddelde wachtduur altijd groter; dit is eerder onderstreept in §8.2.

De variantie in de serviceduren heeft dus een grote invloed op de gemiddelde wachtduur. Goed werken is kennelijk niet alleen snel werken, maar ook met regelmaat werken, anders wordt er toch gemiddeld lang gewacht. Het kortst wordt gewacht als alle klanten evenveel werk van de server vragen (dat kan ook bij andere aankomstpatronen dan volgens een Poissonproces bewezen worden).

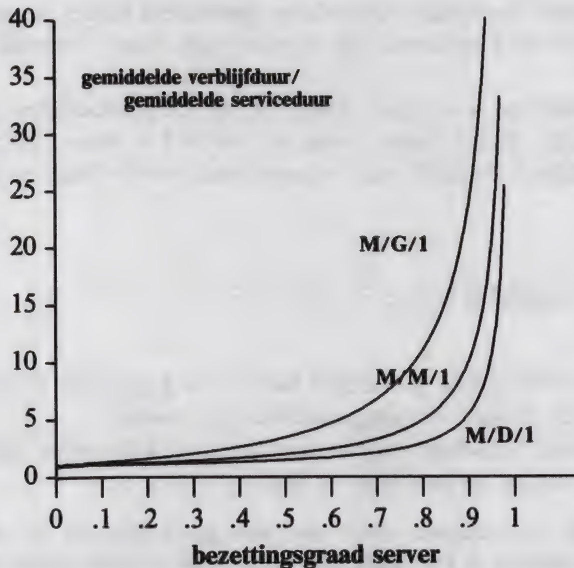
Uit het hoofdstuk BEURTEN EN KANSEN weten we hoe dit komt. Bij spreiding in de serviceduren treden er naast relatief korte serviceduren ook serviceduren op, die veel langer zijn dan gemiddeld. De kans een lange serviceduur aan te treffen kan aanzienlijk verschillen van de kans op het optreden er van. Er moet, als een lange serviceduur wordt getroffen, lang gewacht worden.

15.11. INZICHTEN

De formule van Pollaczek-Khinchin voor de gemiddelde wachtduur in zo algemeen systeem als M/G/1 bij een non-preemptive P-K scheduling, is opvallend eenvoudig. Naast de gemiddelde restduur treedt slechts de opblaasfactor $1/(1-U)$ op.

Bij een systeem dat het niet druk heeft, is U klein en is $1/(1-U)$ dicht bij 1. De gemiddelde wachtduur is dan klein; vrijwel iedereen wordt direkt geholpen.

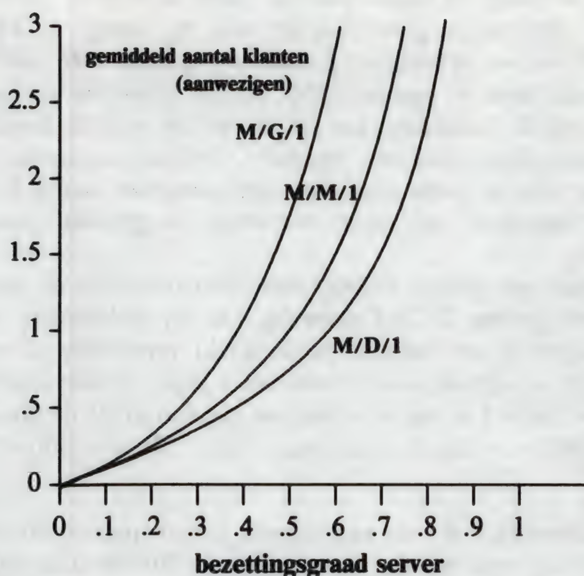
Maar als het systeem het druk krijgt wordt het anders. Als er meer werk per tijdseenheid wordt aangeboden, werk van hetzelfde soort als eerst —met dezelfde verdeling van de serviceduur—, wordt de doorstroom groter en daardoor de bezettingsgraad hoger. Het wachten groeit volgens het hyperbolische verband $1/(1-U)$ uit de opblaasfactor (figuur 15.2-3).



Figuur 15.2. Verhouding van gemiddelde verblijfduur en gemiddelde serviceduur als funktie van de bezettingsgraad voor diverse waarden van de variatiecoefficient C van de serviceduur. M/G/1 systeem met een P-K regeling, C is 2 (M/G/1), 1 (M/M/1), 0 (M/D/1).

Als U toeneemt van 0 naar 1, stelt de opblaasfactor heel lang weinig voor, maar bij een U van ongeveer 60-70% begint een snelle stijging, die uitgroeit tot de waarde 10 bij $U = 90\%$ en de waarde 100 bij $U = 99\%$.

Dit effekt van snel oplopende wachtperikelen doet zich ook voor bij de gemiddelde verblijfduur, het gemiddeld aantal wachtenden en aanwezigen en ook bij de tijd van continu werken. Bij $U = 20\%$ krijgt de server gemiddeld om de



Figuur 15.3. Gemiddeld aantal klanten (aanwezigen) als funktie van de bezettingsgraad voor diverse waarden van de variatiecoefficient C van de serviceduur. M/G/1 systeem met een P-K regeling, C is 2 (M/G/1), 1 (M/M/1), 0 (M/D/1).

$1/(1-0.2) = 1.25$ opdrachten een moment voor zichzelf. Als $U = 90\%$ krijgt hij dit slechts om de 10 opdrachten.

De bezettingsgraad U is in open systemen als M/G/1 en M/M/1 sterk bepalend voor de mate van wachten. Als het systeem het heel druk krijgt gaat U naar 1; dat veroorzaakt een asymptotisch naar oneindig oplopen van de gemiddelde wachtduur. Voor $U = 1$ is de opblaasfactor onbepaald, de server kan het werk niet meer aan, zodat de rij wachtenden groter en groter wordt. In dit randgeval gaat de afleiding niet langer op, er is geen sprake van een stationaire toestand, de rij blijft groeien want klanten blijven maar binnenkomen volgens het Poissonproces.

Soms wordt gesteld dat de opblaasfactor iets zou aangeven dat karakteristiek is voor elk wachttijdensysteem. Er wordt dan op grond van 'wachttijdentheorie' zorgvuldig voor gewaakt dat de bezettingsgraad van servers (CPU, kanalen) niet boven een waarde van 70% (als 'vuistregel') uitkomt, anders zou er excessief worden gewacht. Het beschreven effekt is echter niet karakteristiek voor een wachttijdensysteem op zich. Het is kenmerkend voor de open systemen, maar het doet zich niet voor in gesloten systemen. Dit wordt duidelijk als we bedenken waardoor het wachten ontstaat.

Zowel het aankomstpatroon als het verwerkingspatroon vertonen stochastische fluktuaties. Nu eens komen er veel opdrachten achterelkaar binnen, dan weer weinig; nu eens werkt de server kwiek, want de opdrachten zijn klein, dan weer duurt

het lang omdat er toevallig een lange opdracht wordt afgewerkt. Als U kleiner dan 1 is kan de server zijn zaakjes gemiddeld de baas, hij brengt zelfs bij een bezettingsgraad van 90% en een opblaasfactor van 10 nog altijd 10% van zijn tijd werkeloos door —zonder werk en ogenschijnlijk zonder direkt aantoonbaar nut voor zijn werkgever. Door de fluktuaties kan de server het bij zo'n doorstroom echter gemiddeld niet meer klaarspelen een opdracht vlot na aanbieden te verwerken, meestal kwamen er juist te voren nogal wat opdrachten en waren er zojuist nogal wat lange jobs, waardoor de server achterop is geraakt, enzovoort, dus: verontschuldigingen.

Als de opdrachten met gelijke tussenpozen zouden komen en de serviceduren alle even lang waren (in een D/D/1 systeem), zou het anders zijn. Als U kleiner dan 1 is kan de server de opdrachten steeds direkt verwerken en wordt er nooit gewacht. Zelfs bij de maximale doorstroom van 1 klant per serviceduur, dus bij $U = 1$, geldt dit. De server kan na het afwerken van een klant direkt doorgaan met de nieuwkomer. Maar zo ligt het algemeen niet. De spreidingen veroorzaken wachten.

In een gesloten systeem (§14.8), met een beperkt aantal (potentiële) klanten, zal de voorgaande redenering veel minder hout snijden. Als er daar door toevallige omstandigheden veel opdrachten wachten, komen er ook minder binnen en een server kan veel meer in rust en vree zijn queue afwerken, zonder te moeten constateren dat er steeds weer nieuwe opdrachten bij komen. Een server heeft in een gesloten systeem bij hoge bezettingsgraden veel betere kansen zijn wachtrij gemiddeld kort te houden.

De server in een open systeem als M/G/1 komt zowel in moeilijkheden door spreiding in de serviceduren als door stochastische fluktuaties in het aankomstpatroon. In het M/D/1 systeem zijn er alleen fluktuaties in het aankomstpatroon en ontstaat een wachtrij doordat klanten af en toe 'te dicht op elkaar' binnenkomen. Het wegvallen van de fluktuaties in de serviceduren maakt dat bij de P-K regelingen de gemiddelde wachtduur ten opzichte van negatief exponentieel verdeelde serviceduren wordt gehalveerd (§15.10).

Ook bij de spreiding in het aankomstpatroon zal het zijn: hoe minder spreiding, des te minder wachten.

In computerconfiguraties worden buffers gebruikt om de spreiding en fluktuatie in aan- en afvoer op te vangen. Buffers zijn wachtruimte, die ter beschikking van wachtenden wordt gesteld. Hoe groot een buffer moet zijn hangt sterk van de spreiding in het aantal aanwezigen af. Deze verdeling hebben we niet bepaald; deze is voor het M/M/1 systeem te vinden met de evenwichtsvergelijkingen voor Markovprocessen (§16.2) en in het M/G/1 systeem met een meer geavanceerde analyse.

15.12. SAMENVATTING

De gemiddelde wachtduur in een M/G/1 systeem onder een non-preemptive P-K regeling kan berekend worden met de methodiek van RESTDUREN EN WACHTDUREN. De gemiddelde wachtrestduur, vermenigvuldigd met $1/(1-U)$, levert de gemiddelde wachtduur.

De gevonden wachtpatronen zijn karakteristiek voor open systemen. Als de bezettingsgraad hoog is, wordt er gemiddeld lang gewacht. Een klant in een computerconfiguratie kan vaak niet onderscheiden of hij geholpen wordt of wacht. De server, die met een vaste snelheid werkt, doet zich bij hoge belastingen als traag voor, maar is niet traag doch heeft het druk.

Bij een grote spreiding in de serviceduur (hyper-exponentieel) is de gemiddelde restduur langer dan de gemiddelde serviceduur en wordt er gemiddeld ook lang gewacht.

Het M/M/1 systeem is een speciaal geval van het M/G/1 systeem.

15.13. OPGAVEN

opgave 15.1: tweede moment

Bepaal de gemiddelde wachtduur voor de situatie uit tabel 15.4 door de grootheid $E(T_s^2)$ uit te rekenen en de formule (15.3) toe te passen. De methode is:
Bepaal voor elke component $E(T_s^2)$ uit de variatiecoëfficiënt via $C^2 = VAR(T_s)/E^2(T_s)$ en $VAR(T_s) = E(T_s^2) - E^2(T_s)$ (§7.7). Vermenigvuldig deze uitkomsten met de kans dat de component optreedt, dus met het relatieve aandeel in de doorstroom.

Bereken de variatiecoëfficiënt van de serviceduur. Is de verdeling hypo-exponentieel?

opgave 15.2: transakties en accessen

Een configuratie verwerkt onder andere opdrachten van soort *A* en opdrachten van soort *B*. Bij het verdelen van de software over de disks is er voor gezorgd dat beide soorten samen een disk delen. Naast deze disk gebruiken ze gezamenlijk of apart mogelijk nog andere I/O-devices.

Door de disk worden 20 accessen van soort *A* per sekonde verwerkt en 10 van soort *B*. De accessen van soort *A* worden in 1/5 deel van de gevallen in precies 10 msek. verwerkt, in de andere gevallen in precies 20 msek. De accessen van soort *B* worden in gemiddeld 20 msek. verwerkt, de verdeling van de verwerkingsduur is negatief exponentieel.

Een transaktie van soort *A* genereert gemiddeld 20 accessen op deze disk en een transaktie van soort *B* gemiddeld 15.

De accessen van soort *A* en soort *B* komen binnen volgens een Poissonproces. De afwikkelingsvolgorde bij de verwerking is FCFS. De disk gedraagt zich als een fixed-rate server.

- Bepaal de gemiddelde wachtduur per acces bij de disk (16.82).
- Bepaal de gemiddelde verblijfduur per acces bij de disk voor soort A.
- Wat is voor de transakties van soort A de gemiddelde verblijfduur bij de disk (696.4)?
- Wat is de (overall) gemiddelde verblijfduur bij de disk per transaktie (638.7)?

opgave 15.3: sneller

Er komen ergens twee opdrachten per sekonde binnen volgens een Poissonproces. De verdeling van de serviceduren van de opdrachten is negatief exponentieel. Een opdracht wordt direkt na binnenkomst lukraak ingedeeld bij een van de groepen A en B; een opdracht kan niet van groep veranderen. Groep A wordt verwerkt door de fixed-rate server A met een gemiddelde bedienduur van 0.5 sekonde. Groep B wordt verwerkt door de fixed-rate server B met een gemiddelde bedienduur van 0.25 sekonde. De regeling van de verwerking bij server A en bij server B is LCFS, er is voldoende wachtruimte.

- Het aantal opdrachten dat in 3 sekonden binnenkomt noemen we X. Hoe groot is X gemiddeld? Gedurende geruime tijd wordt X gemeten, daarna wordt van de uitkomsten een histogram gemaakt. Welk deel van de waarnemingen uit het histogram zal kleiner zijn dan een derde van het gemiddelde van X (vrijwel 0.02)?
- Welk deel van de serviceduren van de opdrachten is kleiner dan gemiddeld?
- Bereken de gemiddelde verblijfduren (responsetijden) van de opdrachten (1; 1/3;2/3).
- Bepaal het gemiddeld aantal wachtende opdrachten (7/12).

Men vervangt de servers A en B samen door een server C. Beide groepen opdrachten worden nu door C verwerkt. Server C is ook een fixed-rate server. De afwikkelingsregeling wordt FCFS. Het gemiddeld aantal binnengekomen opdrachten, dat op een lukraak moment nog niet is verwerkt, blijkt voor en na de vervanging hetzelfde te zijn.

- Bepaal de gemiddelde bedienduur voor server C.

opgave 15.4: per job of per klus

Er komen 2 jobs per sekonde binnen volgens een Poissonproces. De jobs worden verwerkt door een server met een vaste verwerkingssnelheid. De gemiddelde serviceduur bij deze server als standaardserver is 1/4 sekonden per job, de verdeling van de serviceduren is negatief exponentieel. De verwerking verloopt FCFS, er is voldoende wachtruimte.

- Bereken de gemiddelde responsetijd (verblijfduur) van de jobs (1/2).
- Bepaal het gemiddeld aantal wachtende jobs, dat een binnenkomende job aantreft. Gebruik geen formule, maar ga uit van de gemiddelde responsetijd (1/2).
- Veronderstel dat de klanten elke job in drie 'klussen' verdelen, die elk het derde deel van de verwerkingstijd van de job nemen. Deze klussen worden apart aangeboden. Bepaal de doorstroom aan 'klussen' en de bezettingsgraad van de server.
- Neem aan dat de klanten het met elkaar voor elkaar krijgen de klussen zonder enig vast patroon op lukrake momenten aan te bieden. Geef in de nieuwe situatie de gemiddelde verblijfduur van een klus (1/6) en van een job (1/2). Bepaal ook het gemiddeld aantal klussen dat op een lukraak moment op verwerking wacht.
- Beantwoord de voorgaande vragen ook als de variatiecoëfficiënt van de serviceduren van de jobs 2 is (7/8, 5/4, 7/24).

opgave 15.5: vertraging bij transport over een lijn

Een cluster controller verbindt een aantal terminals met een mainframe. Tussen cluster controller en mainframe loopt een 4800 bps (600 byte/sek.) lijn (simplex). Tijdens piekuren zijn er 8 terminalisten aktief op de terminals uit het cluster. Ze genereren per uur en per terminal gemiddeld 173 opdrachten voor het mainframe, deze opdrachten zijn gemiddeld 39 bytes lang. Er komen per uur en per terminal gemiddeld 1160 messages van de configuratie terug naar de terminalisten, deze messages zijn gemiddeld 161 bytes lang. Doordat de terminalisten met 'fullscreen' applicaties werken, komen er per opdracht veel bytes terug.

- De 4800 bps lijn wordt als een server opgevat. Bepaal de gemiddelde serviceduur voor het huidige verkeer over deze lijn. Is deze server een load-dependent server? Bepaal de bezettingsgraad van de lijn (0.72).
- Het wachttijdensysteem rond de lijn wordt opgevat als een M/M/1 systeem onder een P-K regeling. Is het redelijk te veronderstellen dat de serviceduren en de tussenpozen bij aankomst negatief exponentieel verdeeld zijn? Wat is de verhouding van de gemiddelde verblijfduur 'bij de server lijn' en de gemiddelde serviceduur (3.57)? Welk deel van de verblijfduur wordt gemiddeld met wachten doorgebracht (U)?
- Een opdracht verblijft gemiddeld 1.0 sekonden in het mainframe. De cluster controller en de controllers die de toegang tot het mainframe mogelijk maken, zijn snel vergeleken met de 4800 bps lijn en veroorzaken geen merkbare vertraging. Geef een afschatting voor de gemiddelde responsetijd voor de terminalisten uit het cluster. Neem aan dat de responsetijd is afgelopen zodra de eerste message terug is van het mainframe.

- Modelleer het wachttijdensysteem rond de lijn ook als een M/G/1 systeem. Welke veranderingen moeten in de berekeningen worden aangebracht, zijn er voldoende gegevens? Neem bijvoorbeeld opdrachten van een vaste lengte en messages waarvan de lengte uniform verdeeld is. Zijn er in de uitkomsten duidelijke verschillen met de M/M/1 aanname (ja)?

16

Eigenschappen M/M/1 resultaten

16.1. INLEIDING

De uitkomsten voor het M/M/1 systeem weerspiegelen de lukrake voortgang in dit wachttijdensysteem. Ze zijn een goede eerste benadering bij veel wachtproblemen uit de computer-prestatieanalyse. Daarom gaan we nader in op de achtergrond van deze uitkomsten. De groep wachttijdensystemen met globaal dezelfde prestaties als het M/M/1 systeem onder P-K speelt een belangrijke rol bij het doorrekenen van computersystemen. Een queuing-netwerk opgebouwd uit systemen van dit type is 'separabel'.

Evenwichtsvergelijkingen volgen uit het principe van 'binnendruppelen is weglekken' uit het hoofdstuk MARKOVKETENS EN EVENWICHTSVERGELIJKINGEN, ze hebben een operationele achtergrond. Deze evenwichtsvergelijkingen worden voor systemen met een continu verloop in de tijd gebruikt om kansen op toestanden te berekenen. Dit laten we hier zien voor het M/M/1 systeem.

16.2. KANS OP AANTAL KLANTEN

Voor het M/G/1 systeem onder een P-K regeling zijn in het voorgaande hoofdstuk de gemiddelde waarden voor de prestatiegrootheden berekend. Als de serviceduren negatief exponentieel verdeeld zijn, kan nog meer informatie worden verkregen: voor het M/M/1 systeem kan de kans een zeker aantal klanten aan te treffen eenvoudig worden bepaald.

Hierbij wordt gebruik gemaakt van het principe van 'binnendruppelen is weglekken', dat voor Markovketens in het hoofdstuk MARKOVKETENS EN EVENWICHTSVERGELIJKINGEN is geïntroduceerd. Dit wat plastisch omschreven procédé beschrijft een basiskenmerk voor elke stationaire toestand: de kans om uit een

toestand te raken is in de stationaire toestand even groot als de kans om in die toestand terecht te komen.

Het is in §11.4 gebruikt om voor een stapsgewijs tijdsverloop de evenwichtsvergelijking $\pi = \pi P$ op te stellen. Voor een continu tijdsverloop gelden soortgelijke overwegingen. De overgang naar de 'echte' kansen kan alleen gemaakt worden als er 'geen geheugen' is. Dat betekent bij een continu tijdsverloop dat overgangen op lukrake momenten plaatsvinden. Men spreekt dan van een *Markovproces*. Een ruime klasse van wachttijdensystemen gedraagt zich als een Markovproces.

Het M/M/1 wachttijdensysteem is zo'n Markovproces. Het intreeproces is een Poissonproces en ook het uittreden van klanten gebeurt, door de P-K regeling en omdat de serviceduren negatief exponentieel verdeeld zijn, lukraak.

Als identificatie voor de toestanden van het systeem wordt het aantal in het systeem aanwezige klanten genomen. Dit is voor het M/M/1 systeem, dank zij het lukrake karakter, een voldoende karakterisering. Er hoeft bijvoorbeeld niet te worden aangegeven hoelang het geleden is dat het systeem idle was, of wanneer de klant die nu door de server verwerkt wordt, in behandeling is genomen.

De toestanden worden aangegeven als toestand$_i$, met i klanten in het wachttijdensysteem. De kans om als lukrake waarnemer het systeem in toestand$_i$ aan te treffen geven we aan met p_i. Dit is een kans op aantreffen, niet op optreden. Het aantal i is positief of nul, er is geen bovengrens omdat de wachtruimte onbeperkt is.

We stappen tijdelijk over op de notatie, die gebruikelijk is bij evenwichtsvergelijkingen voor wachttijdensystemen (§12.2.4, §13.5). De doorstroom wordt aangegeven met λ, er komen per tijdseenheid gemiddeld λ klanten binnen. De gemiddelde serviceduur is $1/\mu$.

De klanten komen volgens een Poissonproces binnen. Volgens de basisrelatie van het Poissonproces is in ieder klein tijdintervalletje van lengte dt de kans op een signaal $\lambda\, dt$, of anders gesteld: de kans op een signaal hangt niet van de tijd af. Er is dus een vaste kans $\lambda\, dt$ dat er in een klein tijdintervalletje van lengte dt een klant binnenkomt.

Door de P-K afwikkeling zijn de tussenpozen bij uittreden verdeeld als de serviceduren, dus negatief exponentieel. Daardoor is er bij een werkende server ook een vaste kans $\mu\, dt$ dat er een klant vertrekt in zo'n tijdintervalletje. De kans op twee van de signalen aankomst en vertrek in een klein intervalletje, is een factor dt kleiner dan de aangegeven kansen op één signaal en verwaarloosbaar (§12.2.3, §12.3.3). Er hoeft daardoor alleen rekening te worden gehouden met overgangen naar de toestanden met een klant meer door aankomst van een klant of met een klant minder door vertrek van een klant.

De kans om in een klein tijdintervalletje *dt* toestand$_i$ met *i* klanten in het systeem te verlaten, is zodoende opgebouwd uit twee stukken: de tijdsduur *dt* maal

$\lambda \times p_i$: de kans p_i op de toestand die verlaten wordt, maal de overgangssnelheid λ van toestand$_i$ naar toestand$_{i+1}$ door aankomst van een klant.

$\mu \times p_i$: de kans p_i op de toestand die verlaten wordt, maal de overgangssnelheid μ van toestand$_i$ naar toestand$_{i-1}$ door vertrek van een klant.

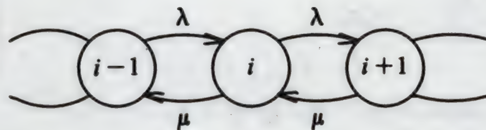
Dit is het 'weglek' deel van de relatie.

De kans om in een klein tijdintervalletje *dt* naar toestand$_i$ over te gaan is *dt* maal de som van

$\lambda \times p_{i-1}$: de kans p_{i-1} op de toestand die verlaten wordt, maal de overgangssnelheid λ van toestand$_{i-1}$ naar toestand$_i$ door aankomst van een klant.

$\mu \times p_{i+1}$: de kans p_{i+1} op de toestand die verlaten wordt, maal de overgangssnelheid μ van toestand$_{i+1}$ naar toestand$_i$ door vertrek van een klant.

Dit is het 'binnendruppel' deel van de gelijkheid.



Figuur 16.1. Binnendruppelen en weglekken in M/M/1 systeem.

Na gelijkstellen van binnendruppelen en weglekken en wegstrepen van *dt*, volgt als evenwichtsrelatie

$$(\lambda+\mu)p_i = \lambda p_{i-1} + \mu p_{i+1}$$

Deze evenwichtsrelatie kan ook operationeel worden afgeleid, analoog aan de herschrijvingen in §11.4 en §13.5, zie §16.2.2.

De vergelijking geldt voor positieve *i*. Voor toestand$_0$ moet de evenwichtsvergelijking apart worden vastgesteld. Er is in die toestand geen weglekken door vertrek van een klant. De evenwichtsrelatie voor toestand$_0$ is

$$\lambda p_0 = \mu p_1$$

De verkregen verzameling evenwichtsvergelijkingen is heel gemakkelijk op te lossen. Allereerst constateren we dat

$$\lambda p_i = \mu p_{i+1}$$

voor alle $i \geqslant 0$ aan de relaties voldoet. De kans op $i+1$ klanten is dus te halen uit de kans op i klanten:

$$p_{i+1} = \frac{\lambda}{\mu} p_i$$

Hier staat een recurrente relatie tussen de kansen op de toestanden. Iedere kans is daarmee terug te brengen tot de kans p_0 op de toestand met 0 klanten. De oplossing is

$$p_i = (\frac{\lambda}{\mu})^i p_0$$

Tenslotte moet de kans p_0 gevonden worden uit de eis dat de som van de kansen 1 is. Dat levert hier —want i is onbegrensd— dat

$$\sum_{i=0}^{\infty} (\frac{\lambda}{\mu})^i = 1/p_0$$

Sommatie van deze meetkundige reeks geeft, mits $\lambda/\mu < 1$

$$p_0 = 1 - \frac{\lambda}{\mu}$$

Dit is geen grote vondst. De kans p_0 op geen klant in het systeem is de tijdgemiddelde kans de server zonder werk te vinden, dus $1-p_0$ is de bezettingsgraad U. Er staat de basisrelatie doorstroom $= U/E(T_s)$ of $\lambda = \mu U$ in de nieuwe notatie. De kans dat er i klanten in het systeem zijn, is daarmee

$$p_i = U^i(1-U)$$

De verhouding λ/μ wordt in een open systeem als M/M/1 ook wel de *verkeersdichtheid* genoemd. In de stationaire situatie is deze altijd kleiner dan 1 (§14.5). We zien hier dat anders de som van de kansen niet tot 1 convergeert.

Bij deze aanpak worden de prestatiegrootheden uitgerekend via het gemiddeld aantal aanwezigen:

$$\textit{gem. aantal aanwezigen} = \sum_{i=1}^{\infty} i\, p_i$$

$$= (1-U) \sum_{i=1}^{\infty} i\, U^i$$

Sommatie geeft als uitkomst $U/(1-U)$, wat we al eerder via de relatie van Pollaczek-Khinchin in §15.9 hebben verkregen.

Op dezelfde manier wordt de variantie in het aantal aanwezige klanten berekend. De uitkomst is $U/(1-U)^2$.

De andere prestatiegrootheden volgen uit het gemiddeld aantal aanwezigen met de betrekkingen uit §14.7.4 (§15.9).

gem. verblijfduur	=	*gem. serviceduur / (1 − bezettingsgraad)*
gem. wachtduur	=	*gem. serviceduur × bezettingsgraad / (1 − bezettingsgraad)*
gem. aantal aanwezigen	=	*bezettingsgraad / (1 − bezettingsgraad)*
gem. aantal wachtenden	=	*bezettingsgraad*^{2} */ (1 − bezettingsgraad)*
kans op i klanten	=	*(1 − bezettingsgraad) × bezettingsgraad*^{i}
variantie aantal aanwezigen	=	*bezettingsgraad / (1 − bezettingsgraad)*^{2}

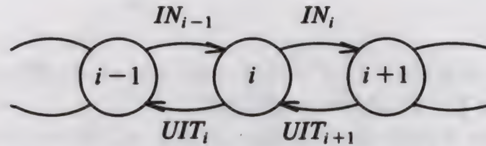
De evenwichtsvergelijkingen leiden dus tot dezelfde resultaten voor de gemiddelden als de aanpak via de wachtrestduur en de relatie van Little. De opbrengst van de evenwichtsvergelijkingen is rijker, doordat de kans om een bepaald aantal klanten aan te treffen kan worden berekend. Dit is van nut als er informatie over de verdeling van het aantal (wachtende) klanten gewenst is.

16.2.1. voorbeeld: M/M/1 kansen

Er komen bij een server op lukrake momenten klanten binnen, waarvan de opdrachten op lukrake momenten zijn afgelopen. De bezettingsgraad van de server is 2/3. De server is 1/3 deel van de tijd vrij (p_0), hij is 2/9 deel van de tijd met een klant bezig zonder dat er een andere klant wacht (p_1). Een binnenkomende klant treft met een kans 4/27 één wachtende voorganger aan (p_2). Het gemiddeld aantal klanten in het systeem is 2 met een standaarddeviatie van $\sqrt{6}$.

16.2.2. evenwichtsvergelijking (operationeel intermezzo)

In de meetduur zal de overgang van toestand$_i$ naar toestand$_{i+1}$ door het binnenkomen van een klant in totaal IN_i maal geregistreerd worden (de afspraken voor het tellen zijn als in §3.4). De overgang naar toestand$_{i-1}$ door vertrek van een klant zal UIT_i maal worden geteld. Het systeem zal tijdens $WERK_i$ t.e. in toestand$_i$ zijn.



Figuur 16.2. Binnendruppelen en weglekken in M/M/1 systeem. Tellingen.

De behoudsrelatie voor toestand$_i$ (§11.4)

$$IN_i + UIT_i = IN_{i-1} + UIT_{i+1}$$

kan met operationele schatters herschreven worden. De rechterkant is na delen door *MEETDUUR* (mits $WERK_{i-1}$ en $WERK_{i+1}$ ongelijk nul)

$$(IN_{i-1} / WERK_{i-1})(WERK_{i-1} / MEETDUUR) \\ + (UIT_{i+1} / WERK_{i+1})(WERK_{i+1} / MEETDUUR)$$

De operationele schatter $IN_i / WERK_i$ schat de snelheid van aankomst in toestand$_i$ (§13.5, §14.4.2). De operationele schatter $WERK_i / UIT_i$ schat de gemiddelde bedienduur in toestand$_i$ (§14.4.3, de schatter voor de gemiddelde bediensnelheid in toestand$_i$ is $UIT_i / WERK_i$). De operationele schatter voor de kans toestand$_i$ aan te treffen is $WERK_i / MEETDUUR$ (§3.4). Als de snelheid van aankomst en de gemiddelde bedienduur niet van de omstandigheden afhangen, en speciaal niet van i, geeft overgang naar de 'echte' overall waarden voor het rechterlid (vergelijk §13.5)

$$\lambda p_{i-1} + \mu p_{i+1}$$

Het linkerlid kan analoog worden herleid. De behoudsrelatie levert daarmee

$$(\lambda + \mu) p_i = \lambda p_{i-1} + \mu p_{i+1}$$

16.3. EVENWICHTSVERGELIJKING EN SCHEDULING

De gevonden kansen voor een M/M/1 systeem onder een P-K regeling gelden ook als de regeling preemptive is. De redenering loopt als volgt.

Bij zo'n regeling moet een klant die door de server geholpen wordt, soms wijken voor een ander. We verwaarlozen de overhead die daardoor ontstaat. Vlak voor een gedwongen onderbreking is er de vaste kans $\mu\, dt$ dat de klant de server binnen een tijdinterval van lengte dt afgewerkt verlaat. Na de onderbreking werkt de server voor de nieuwe klant. De kans dat een klant vertrekt is niet veranderd, want ook de nieuwe klant heeft een serviceduur, die getrokken is uit de negatief exponentiële verdeling met als gemiddelde $1/\mu$. Als de verdreven klant naderhand weer op de server komt, gaat de verwerking op deze voet verder; de informatie dat hij al een zekere tijd is verwerkt, heeft bij negatief exponentieel verdeelde serviceduren geen betekenis. Door de gedwongen onderbrekingen verandert het uittreeproces bij negatief exponentieel verdeelde serviceduren dus niet en de kans op een bepaald aantal klanten in het systeem is niet anders dan bij een non-preemptive regeling.

In een M/M/1 systeem maakt dus de verdeling van de serviceduur dat het uittreeproces geheugenloos opereert. De afwikkelingsregeling blijkt in een M/G/1 systeem ook daarvoor te kunnen zorgen. Het is mogelijk deze zo te kiezen dat klanten op (vrijwel) lukrake momenten uittreden, wat de verdeling van de serviceduren ook is. Door zo'n scheduling worden samenhangen —als die er zijn— afgebouwd!

Het zijn geen exotische regelingen die dit presteren. Het zijn gewoon de ROUND ROBIN regeling bij korte time slice, dus de PS (PROCESSOR SHARING) regeling en de regeling PREEMPTIVE LCFS. Voor deze regelingen is bewezen dat ze voor M/G/1 systemen de M→M eigenschap hebben: aankomstproces en uittreeproces (in real time) zijn Poissonprocessen. Deze eigenschap delen ze met het M/M/1 systeem onder P-K en het M/G/1 systeem met een infinite server (§14.9.1). Deze groep regelingen is erg belangrijk. In queuing-netwerken zorgen ze door hun eigenschap van 'local balance' voor het ontstaan van eenvoudige 'separeerbare' uitdrukkingen van de kansverdelingen (§14.9.2).

De ROUND ROBIN regeling is ontwikkeld om het hulpmiddel server 'eerlijker' over de klanten te verdelen dan het geval is bij een FCFS regeling, waar af en toe klanten met een lange job de afwikkeling ophouden. Deze democratisering blijkt dus *randomisering* te bevorderen, waarmee bedoeld wordt dat het uittreden (bij benadering) lukraak gebeurt. De 'eerlijker' afwikkeling schudt de zaken door elkaar. Het is heel plezierig voor de prestatie-analyse dat dit verschijnsel zich steeds voordoet als iets (een protocol, een device dat gezamenlijk gebruikt wordt) geschikt moet blijven voor uiteenlopend gebruik. Vandaar weer het belang van het Poissonproces: veel gebeurt ook om redenen van 'fairness' op vrijwel lukrake momenten.

De PS regeling is karakteristiek voor een grote groep regelingen die met time slices werken. Het 'randomiserende' effekt ontstaat doordat de klant na elk einde van zijn time slice met een andere groep klanten te maken krijgt. Elke klant

maakt per time slice een wachtronde. Iedere ronde staan er andere klanten met andere serviceduren in de wachtrij, want er verdwijnen klanten onder de hand en er komen nieuwe binnen. Bij FCFS daarentegen is het hele wachtlot van de klant vastgelegd bij binnenkomst, er verandert voor hem niets tijdens zijn wachtduur.

De afleiding van de evenwichtsvergelijkingen met het principe van 'binnendruppelen is weglekken' is dus ook voor willekeurige verdelingen van de serviceduren juist, mits de afwikkelingsregeling randomiserend werkt. De in de voorgaande paragraaf gevonden evenwichtsvergelijkingen

$$\lambda p_i = \mu p_{i+1}$$

en daarmee de kans op i klanten in het systeem

$$p_i = (\lambda/\mu)p_{i-1} = U^i(1-U)$$

blijken te gelden voor

- M/M/1 systemen onder een P-K regeling, preemptive of non-preemptive.
- M/G/1 systemen onder PS en onder preemptive LCFS.

Dezelfde verdeling van de aantallen betekent volgens Little ook dezelfde gemiddelde verblijfduur of responsetijd. Al deze regelingen zijn gelijkwaardig op de punten verdeling van aantal aanwezigen en gemiddelde verblijfduur, de globale prestatiegrootheden verschillen niet.

De non-preemptive LCFS regeling wordt gebruikt bij het werken met stacks, de preemptive LCFS regeling bij het 'transparant' stacken van interrupts. Bij het 'stacken' van klanten zijn de kansen op een bepaald aantal aanwezigen niet anders dan bij het FCFS 'queuen' van klanten in een M/M/1 systeem.

Merk op dat een randomiserende regeling als PS (ROUND ROBIN) soms een langere gemiddelde verblijfduur oplevert dan een P-K regeling als FCFS. Dit is het geval als de serviceduur hypo-exponentieel is verdeeld (variatiecoëfficiënt in de serviceduur kleiner dan 1, §15.3.4 (15.5), §15.10).

16.3.1. verdeling verblijfduur en scheduling

Helaas eindigt dit overzicht met een tegenvaller: de verdelingen rond de gemeenschappelijke gemiddelde waarde voor de verblijfduur zijn voor de aangegeven regelingen sterk verschillend. Een LCFS regeling bijvoorbeeld geeft een veel grotere spreiding in de verblijfduren dan een FCFS regeling.

Hier stuiten we op iets dat ons niet mag verrassen, maar dat zeker teleurstelt: hoe precies ook de kennis over de kansen op de verschillende toestanden is, de verdeling over de verblijfduren kan er niet mee bepaald worden. Met de kansen kan

alleen de gemiddelde verblijfduur via Little worden berekend. Alle bovenstaande afwikkelingsregelingen geven totaal andere verdelingen van de verblijfduren.

Bij operatingsystemen zegt men soms dat een regeling met een grote spreiding in de responsetijd aan 'starvation' lijdt, doordat sommige klanten pas na lang wachten geholpen worden. De mate van 'starvation' is voor de aangegeven regelingen niet gelijk.

Om de verdeling van de verblijfduren te berekenen moet de bedieningsregeling dus in detail bekend zijn. Dan pas kan men —meestal met vrij ingewikkelde mathematische technieken— uitspraken doen over deze verdeling. Het voorspellen van de verdeling van de verblijfduren zal altijd veel lastiger zijn dan het redelijk voorspellen van bezettingsgraden, doorstromen, gemiddelde responsetijden en kansen op toestanden.

16.4. GEMIDDELDE VERBLIJFDUUR IN EEN OPEN QUEUING-NETWERK

In *separabele queuing-netwerken* verloopt de verwerking bij elke server uit het netwerk volgens een van de vormen uit de voorgaande paragraaf, dus overeenkomstig P-K, PS of PREEMPTIVE LCFS. Bij een P-K regeling moeten de serviceduren per acces negatief exponentieel verdeeld zijn; wanneer er onderscheid wordt gemaakt tussen verschillende soorten werk mag de gemiddelde serviceduur per acces niet verschillen naar gelang het type werk. Deze beperkingen gelden niet voor de 'randomiserende' regelingen en evenmin voor de infinite servers uit het netwerk. Veel computersystemen gedragen zich in eerste benadering als separabele queuing-netwerken.

De voor het open M/M/1 systeem gevonden 'opblaas'waarde voor de gemiddelde verblijfduur blijkt algemeen geldig te zijn in alle *open separabele queuing-netwerken*. De gemiddelde verblijfduur bij elke server uit het net, en daarmee de totale verblijfduur in het net, kan eenvoudig berekend worden.

We illustreren dit weer aan het voorbeeld van de usercommando's. Tabel 9.1 uit §9.9.3 (of tabel 14.3) specificeert het open queuing-netwerk, dat een model van het 'low level' zou kunnen zijn (§14.9). Er wordt aangenomen dat de servers met vaste snelheid werken. Voor het berekenen van de gemiddelde serviceduren per opdracht kan het in het algemeen nodig zijn de relatie $\pi = \pi \mathbf{P}$ op te lossen (§9.9.2). De gemiddelde totale verblijfduur in het queuing-netwerk en de gemiddelde verblijfduur bij elk van de servers apart staan in tabel 16.1 en 16.2. Merk op dat spreidingen geen rol spelen, dit in tegenstelling tot het M/G/1 systeem onder een P-K regeling.

Het betrokken deel van het 'low level' zal vaak als een gesloten systeem moeten worden beschouwd, omdat het toelatingsbeleid van het operatingssysteem er voor zorgt dat de multiprogrammeringsgraad niet sterk varieert (§14.9). Omdat de gemiddelde wachtduur in een gesloten systeem korter is dan in het overeenkomstige open systeem (§15.11) zijn de verkregen uitkomsten pessimistische afschattingen voor de responsetijd.

1	2	3	4	5
device	ser-vice-duur	bezet-tings-graad	op-blaas-factor	ver-blijf-duur
CPU	10	0.25	1.33	13.3
paging	10	0.25	1.33	13.3
I/O-1	10	0.25	1.33	13.3
I/O-2	20	0.50	2	40
I/O-3	20	0.50	2	40
I/O-4	30	0.75	4	120

Tabel 16.1. Verwerking van de usercommando's (tabel 9.1). Modellering als open systeem, separabel. Tijden in msek. Gemiddelde verblijfduren bij een doorstroom van 75% van de maximale doorstroom van 33.3 opdr./sek. Knelpunt bij I/O-4. De doorstroom is 25 opdrachten/sek. Kolom 2: gemiddelde serviceduur per opdracht. Kolom 5: gemiddelde verblijfduur.
Een opdracht verblijft gemiddeld gedurende $3 \times 13.3 + 2 \times 40 + 120 = 240$ msek. in het queuing-netwerk. Verblijf opgebouwd uit vele bezoeken. Langste gemiddelde verblijfduur bij I/O-4. De gemiddelde (totale) responsetijd bij verwerking door het netwerk is 240 msek.

1	2	3	4	5	6
device	ser-vice-duur	*U* ander werk	*U* in totaal	op-blaas-factor	ver-blijf-duur
CPU	10	0.60	0.85	6.67	66.7
paging	10	0.08	0.33	1.5	15
I/O-1	10	0.70	0.95	20	200
I/O-2	20	0	0.50	2	40
I/O-3	20	0	0.50	2	40
I/O-4	30	0	0.75	4	120

Tabel 16.2. Hinder van ander werk (tabel 9.2). Modellering, ook van het andere werk, als open systeem. Tijden in msek. Gemiddelde verblijfduren bij een doorstroom van 83.3% van de maximale doorstroom van 30 opdr./sek. Knelpunt bij I/O-1. De doorstroom is 25 opdr./sek. Kolom 3: bezettingsgraad ander werk. Kolom 4: totale bezettingsgraad.
Langste gemiddelde verblijfduur bij I/O-1. Een opdracht verblijft gemiddeld gedurende 481.7 msek. (gemiddelde responsetijd) in het queuing-netwerk.

16.5. SAMENVATTING

We sluiten hier onze oriëntatie rond de wachttijdensystemen af. Het eenvoudigste wachttijdensysteem levert uitkomsten op, die ook buiten de strikte context van het model van betekenis zijn. De resultaten voor het M/M/1 systeem zijn een veel gebruikte eerste benadering voor wachtproblemen uit de praktijk van de prestatie-analyse.

16.6. OPGAVEN

opgave 16.1: vervolg transakties en accessen (opgave 15.2)

De verwerking van de soorten A en B wordt opnieuw bekeken, maar nu niet voor verwerking door een disk onder FCFS, maar door een CPU onder ROUND ROBIN met een korte time slice (PS, §16.3). Dit is de enige wezenlijke wijziging, de verwerkingsduren en de verdelingen blijven hetzelfde.

- Bepaal de (overall) gemiddelde verblijfduur bij de CPU per transaktie (763.6).
- Idem die voor soort A en soort B afzonderlijk (818.2, 681.8).

opgave 16.2: opblaasfactor

Systeembeheerders redeneren als ze de vertraging bij een server (processor) berekenen vaak als volgt.

Wanneer de server niet belast wordt kan hij een zeker aantal cycles per sekonde maken. Als hij —bijvoorbeeld— een kwart van de tijd belast wordt, heeft hij maar driekwart van dit aantal cycles per sekonde over. Daardoor is de server onder die omstandigheden een factor 4/3 trager.

- Laat zien dat hiermee de opblaasfactor beschreven wordt.
- De opblaasfactor komt voor in het M/G/1 en het M/M/1 systeem en bij separabele queuing-netwerken. Geef voor deze gevallen commentaar op de redenering.

opgave 16.3: wachtduur voor wachtenden

Klanten die de server vrij vinden worden direkt geholpen, voor hen is de wachtduur nul. Dit drukt de 'overall' gemiddelde wachtduur, die daardoor een verkeerde indruk kan geven van de hinder voor klanten die moeten wachten. Daarom kan het beter zijn de gemiddelde wachtduur op te geven voor de klanten met een wachtduur ongelijk aan nul.

- Geef deze voor een aantal systemen en regelingen (factor $1/U$ groter dan de gemiddelde wachtduur). Bespreek de gecombineerde invloed van deze factor en de opblaasfactor.

17

Literatuur

17.1. BOEKEN

Het aantal boeken over Computer Performance Evaluation neemt gestaag toe. We noemen er enige, die al wat langer op de markt zijn. Samen geven ze een overzicht van wat er met de moderne theorie mogelijk is (§17.3).

Uitstekend geschikt als handleiding voor het modelleren is 'Quantitative System Performance' van LAZOVSKA, ZAHORJAN, GRAHAM en SEVCIK. Dit geeft de algoritmen om computermodellen door te rekenen en past deze toe zonder in mathematische details te treden. Het boek van GELENBE en MITRANI 'Analysis and synthesis of computer systems' geeft de nieuwe ontwikkelingen in mathematisch perspectief. SAUER en CHANDY 'Computer systems performance modeling' richt zich op het relatieve belang van de diverse algoritmen. Een encyclopedisch overzicht van analytische resultaten is te vinden in het 'Computer Performance Modeling Handbook', editor S.S. LAVENBERG. MITRANI 'Modelling of computer and communication systems' geeft in kort bestek een helder beeld van theorie en toepassingen.

Al iets ouder, uit 1978, is het erg toegankelijke boek van KOBAYASHI: 'Modeling and analysis: an introduction to system performance evaluation methodology'. De klassieker op performancegebied is sinds jaar en dag KLEINROCK's 'Queueing systems' (deel I en II).

Van de hand van FERRARI is een wat langdradig, maar nuttig boek over alle problemen bij het opzetten, in kaart brengen en valideren van een onderzoek naar de performance van een computerconfiguratie: 'Computer systems performance evaluation'. Een vernieuwde versie van dit boek werd uitgebracht als 'Measurement and tuning of computer systems' door FERRARI, SERAZZI en ZEIGNER. Ook LEUNG behandelt het modelleren van devices.

Een modern performanceboek, opgebouwd rond de 'klassieke' mathematische technieken, is TRIVEDI: 'Probability and Statistics with reliability, queuing, and computer science applications'. Soortgelijk, maar veel oppervlakkiger, is ALLEN: 'Probability, statistics and queueing theory with computer science applications'.

Er zijn erg veel boeken over de klassieke wachttijdentheorie. Het standaard leerboek van COOPER 'Introduction to queueing theory' sluit goed aan bij de gedachtenwereld van de operationele analyse.

Recente boeken op het gebied van de computernetwerken besteden veel aandacht aan de theorie en toepassingen van prestatie-analyse; een aantal richten zich er speciaal op. Het boek van HAMMOND en O'REILLY bijvoorbeeld geeft een compleet beeld, STUCK en ARTHURS behandelen elementaire afgrenzingen.

17.2. TIJDSCHRIFTEN

De computer-prestatieanalyse heeft raakvlakken met vrijwel alle deelgebieden van de informatica. Artikelen over prestatiegericht onderzoek worden dan ook opgenomen in vakbladen die zich op specifieke deeldisciplines van de informatica richten. De laatste jaren manifesteert de performance-analyse zich daarnaast als een zelfstandige discipline. Er is een gevestigd eigen vakblad, dat nogal mathematisch gericht is: *Performance Evaluation* onder redactie van M. REISER.

Wie zich jaarlijks op de hoogte wil stellen van de vooruitgang in de 'kunst', moet de verslagen lezen van het SIGMETRICS congres van de 'special interest group on measurement and evaluation' van de ACM (Association for Computing Machinery) (§17.3). Verslagen van metingen aan commerciële configuraties zijn te vinden in de proceedings van de congressen van de Computer Management Groups (CMG).

17.3. REFERENTIES

Van de meer specifieke literatuur worden per hoofdstuk enkele karakteristieke voorbeelden genoemd. Ook de bron van inspiratie voor een voorbeeld of opgave staat aangegeven.

Boeken:

LAZOVSKA, E.D., ZAHORJAN, J., SCOTT GRAHAM, G. & SEVCIK, K.C. : *Quantitative System Performance (Computer system analysis using queueing network models).* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1984.

GELENBE, E. & MITRANI, I. : *Analysis and synthesis of computer systems.* Academic Press, New York, 1980.

SAUER, C.H. & CHANDY, K.M. : *Computer systems performance modeling.* Prentice Hall, Inc., Englewood Cliffs, N.J., 1981.

LAVENBERG, S.S. (EDITOR) : *Computer Performance Modeling Handbook.* Academic Press, New York, 1983.

MITRANI, I.: *Modelling of computer and communication systems.* Cambridge University Press, Cambridge, 1987.

KOBAYASHI, H.: *Modeling and analysis: an introduction to system performance evaluation methodology.* Addison Wesley Publ. Co., Reading, Mass., 1978.

KLEINROCK, L.: *Queueing Systems* (vol. I, theory and vol. II, computer applications). John Wiley and Sons, Inc., New York, 1975, 1976.

FERRARI, D.: *Computer Systems Performance Evaluation.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1978.

FERRARI, D., SERAZZI, G. & ZEIGNER, A.: *Measurement and tuning of computer systems.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1983.

LEUNG, C.H.C.: *Quantitative analysis of computer systems.* John Wiley and Sons, Inc., New York, 1988.

TRIVEDI, K.S.: *Probability and statistics with reliability, queuing, and computer science applications.* Prentice Hall, Inc., Englewood Cliffs, N.J., 1982.

ALLEN, A.O.: *Probability, statistics, and queueing theory with computer science applications.* Academic Press, New York, 1978.

COOPER, R.B.: *Introduction to queueing theory.* Edward Arnold, London, 1981 (second edition).

HAMMOND, J.L. & O'REILLY, P.J.P.: *Performance analysis of local computer networks.* Addison Wesley Publ. Co., Reading, 1986.

STUCK, B.W. & ARTHURS, E.: *A computer and communications network performance analysis primer.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

Tijdschriften:

PERFORMANCE EVALUATION, an international journal, editor-in-chief REISER, M. - published by Elsevier Science Publishers B.V. (North Holland), Amsterdam.

UK CMG. - Annual Conferences, 113 Queens Road, Reading, UK.

Operationele analyse:

BUZEN, J.P.: 'Fundamental operational laws of computer system performance'. - *Acta Informatica 7,* 2 (1976), 167-182.

DENNING, P.J. & BUZEN, J.P.: 'Operational Analysis of queueing networks'. - *Proceedings of the 3rd Int. Symp. on Measuring, Modelling and Evaluating Computer Systems,* organised by GMD, Bonn-Bad Godesberg, Beilner, H. & Gelenbe, E. (eds.), North Holland Publ. Co. (1977), 151-172.

DENNING, P.J. & BUZEN, J.P.: 'The Operational Analysis of queueing network models'. - *Computing Surveys 10,* 3 (september 1978), 225-261.

BUZEN, J.P. & DENNING, P.J.: 'Measuring and calculating queue length distributions'. - *IEEE Computer 13,* 4 (april 1980), 33-44. Ook in:
'Operational treatment of queue distributions and mean-value analysis'. - *Computer Performance 1,* 1 (juni 1980), Butterworth & Co (Publishers), IPC Science and Technology Press, Guilford, 6-15.

BRUMFIELD, J.A. & DENNING, P.J.: 'Operational state sequence analysis'. - *Performance '83, Proceedings of the 9th Int. Symp. on Computer Performance Modelling, Measurement and Evaluation,* IFIP Working Group 7.3, Maryland, (mei 1983), North Holland Publ. Co. (1983), 269-283.

DALLERY, A. & DAVID, R.: 'Some new results on operational analysis'. - *Performance '84, Proceedings of the 10th International Symposium on Computer Performance (december 1984),* Parijs, North Holland Publ. Co. (1984), 119-134.

KOWALK, W.: 'An operational view of renewal theory'. - *Proceedings of the 1983 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems,* special issue Performance Evaluation Review (ACM-Sigmetrics quarterly publ.), ACM, New York (1983), 130-137.

« Inleiding »

FERRARI, D.: 'Considerations on the insularity of performance evaluation'. - *IEEE Transactions on Software Engineering SE-12,* 6 (juni 1986), 678-683.

DENNING, P.J.: 'Performance analysis: experimental computer science at its best' (ACM President's Letter). - *Communications of the ACM 24,* 11 (november 1981), 725-727.

BASKETT, F., CHANDY, K.M., MUNTZ, R.R. & PALACIOS, F.G.: 'Open, closed and mixed networks of queues with different classes of customers'. - *Journal of the ACM 22,* 2 (1975), 248-260.

REISER, M.: 'Mean-value analysis and convolution method for queue-dependent servers in closed queueing networks'. - *Performance Evaluation 1,* 1 (1981), 7-18.

LAVENBERG, S.S.: 'A perspective on queueing models of computer performance'. - *Performance Evaluation 10,* 1 (1989), 53-76 (een historisch overzicht van het analytisch modelleren).

« Achtergronden »

BEIJNVOORT, H.J.: 'Monitoring van computersystemen'. - *Prestatie-analyse en Capaciteitsmanagement van Computersystemen.* Symposium georganiseerd door de Werkgroep Computer Prestatie Evaluatie, Afdeling/Sectie Informatietechniek (ASI), Amersfoort (ASI Syllabus, april 1983), 23-36.

PAANS, RONALD: - *A close look at MVS systems: Mechanisms, performance and security.* Elsevier Science Publishers B.V., Amsterdam (1986), (handelsuitgave proefschrift Technische Hogeschool Delft).

« Inleiding operationele analyse »

VAN DER VEEN, B.: *Een operationele analyse van enige wachtmodellen.* Eindhoven, Proefschrift Technische Hogeschool, 1975 (een voorloper).

« De relatie van Little »

LITTLE, J.D.C.: 'A proof of the queueing formule $L = \lambda W$'. - *Operations Research 9,* 3 (1961), 383-387 (het meest gerefereerde artikel uit de computerprestatieanalyse).

« Responsetijdrelaties »

AGRAWAL, S.C., BUZEN, J.P. & THAREJA, A.K.: 'A unified approach to scan time analysis of token rings and polling networks'. - *Proceedings of the 1984 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems,* special issue Performance Evaluation Review (ACM-Sigmetrics quarterly publ.) *12,* 3, ACM, New York, (augustus 1984), 176-185 [opgave 5.4].

« Responsetijd en gebruikers »

TURNER, R., SCHRIESHEIM, J. & MITRA, I.: 'Performance of a DECnet based disk block server'. - *zie voorgaand*, 96-104 [opgave 6.5].

KLEINROCK, L.: 'Distributed systems'. - *Communications of the ACM 28*, 11 (november 1985), 1200-1213.

SHNEIDERMAN, B.: 'Response time and display rate in human performance with computers'. - *Computing Surveys 16*, 3 (september 1984), 265-285 [§6.3].

« Restduren en wachtduren »

BRANDWAJN, A.: 'Models of DASD subsystems with multiple access path: A throughput-driven approach'. - *IEEE Transactions on Computers C-32*, 5 (mei 1983), 451-463.

BERETVAS, TH.: 'DASD performance analysis using modeling'. - *Proceedings ECOMA-13 conference*, Zurich, (oktober 1985), 47-56 [§8.9]

WIJBRANDS, R.J.: *Queuing network models and performance analysis of computer systems.* Eindhoven, Proefschrift Technische Universiteit, 1988, (hoofdstuk 6).

WOLF, J.: 'The placement optimization program: a practical solution to the disk file assignment problem'. - *Proceedings of the 1989 ACM Sigmetrics and Performance '89 International Conference on Measurement and Modeling of Computer Systems*, special issue Performance Evaluation Review (ACM-Sigmetrics quarterly publ.) *17*, 1, ACM, New York, (mei 1989), 1-10.

« Ergodische Markovketens »

FELLER, W.: *An introduction to probability theory and its applications.* Vol. I, John Wiley and Sons, Inc., New York, 1968 (third edition).

« Eigenschappen Markovketens »

DE J. PEREZ-DAVILA, A. & DOWDY, L.W. 'Parameter interdependencies of file placement models in a Unix system'. - *Proceedings of the 1984 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, special issue Performance Evaluation Review (ACM-Sigmetrics quarterly publ.) *12*, 3, ACM, New York, (augustus 1984), 15-26 [§10.4.2].

« Markovketens en evenwichtsvergelijkingen »

KOUVATSOS, D.D. : 'A maximum entropy queue length distribution for a G/G/1 finite capacity queue'. *Performance '86 and ACM Sigmetrics 1986, Joint Conference on Computer Performance Modelling, Measurement and Evaluation*, special issue Performance Evaluation Review (ACM-Sigmetrics quarterly publ.) *14*, 1, ACM, New York, (mei 1986), 224-236 (informatietheorie).

« Signalen en Poissonproces »

ÇINLAR, E.: *Introduction to stochastic processes.* Prentice-Hall, Englewood Cliffs, N.J., 1975.

KARLIN, S. & TAYLOR, H.M.: *A first course in stochastic processes.* Academic Press, New York, 1975 (second edition).

« Het M/G/1 systeem »

PAANS, R.: 'MVS TSO network workload characterization and modelling'. - *Computer Performance 4,* 4 (december 1983), Butterworth & Co (Publishers), IPC Science and Technology Press, Guilford, 229-337 [opgave 15.5].

Register

T

U

V

De auteur is docent Software Engineering aan de HIO-Enschede, de opleiding Hoger Informatica Onderwijs aan de Hogeschool Enschede. Hij begeleidt afstudeer-researchprojecten en geeft les in prestatie-analyse, operatingsystemen en vertalerbouw. Vanaf 1978 is hij lid van de werkgroep 'Computer Prestatie Evaluatie' van het ASI/NGL.

De computer-prestatieanalyse is een belangrijk bestanddeel van de moderne informatica. Het doel van dit boek is vooral redeneringen te onderbouwen die vanuit de dagelijkse ervaring als 'gezond verstand' naar voren worden gebracht bij het beoordelen van de prestaties van computerconfiguraties. De basisprincipes die daarbij terloops en verspreid worden gehanteerd, worden hier vanaf het begin systematisch ontwikkeld en rechtstreeks toegepast.

In dit boek worden een aantal universele wetten en overwegingen behandeld. Het zijn de relatie van Little en de responsetijdrelatie, het verband tussen de kans dat een toestand optreedt en de kans dat de toestand wordt aangetroffen, de beschrijving van toestandsovergangen zonder op de invloed van het verleden te letten: de Markovketens en de schat aan informatie die beschikbaar is als iets lukraak plaatsvindt: het Poissonproces. Met deze basisrelaties wordt de gemiddelde responsetijd in de bekendste klassieke wachttijdensystemen berekend.

ISBN 90 6233 250 1
NUGI 852